

Acknowledgement:

Before presenting my work, I would like to thank all the people who contributed to the success of my internship and who helped me in the drafting of this report. I would like to extend my deep and sincere thanks to my teacher and mentor Mr. Abdekrim Mars, who accepted to lead this work, for his invaluable advice, his continuous encouragement, his high availability, his enormous patience, and the precious time he reserved me throughout the realization of my project. Then, I would like to thank Mobilite family members for your loyalty and hard work. Our success depends on the quality of our staff members and we recognize and appreciate your contributions and achievements. Thank you all. My thanks also go to all my teachers, my classmates, as well as to all my friends who helped me by their presence through all years by being the good friends as they are. . Finally, I would like to thank all the ones who made these years at our dear Esprims for the years full of good memories without forgetting all our teachers and all those who provided us with the technical and scientific means to accomplish this work in favorable conditions

Contents

1	General introduction:	1
2	Project studies	2
2.1	Company presentation:	2
2.2	Study of the existing	2
2.3	Problematical:	3
2.4	Solution:	3
2.4.1	Existing solution:	3
2.4.2	Proposed solution:	3
2.5	Specification of requirement	4
2.6	Project management methodology	4
2.6.1	Presentation of Scrum	4
2.6.2	Roles in SCRUM	5
3	Analysis and specification of need	7
3.1	Analysis of need	7
3.1.1	Identification of actors	7
3.1.2	Functional need	7
3.1.3	Non functional need	8
3.2	Product backlog	8
3.3	Application architecture	9
3.4	Sprint planification	10
3.5	Modeling of functional needs	10
3.5.1	Global use case Diagram	10
3.5.2	Class diagram	11
4	Sprint0”Technical study”	13
4.1	Development Environment and technical choice	13
4.1.1	Hardware environment	13
4.1.2	Development environment	13
4.1.2.1	Modeling tool	13
4.1.2.2	Integrated development environment	14
4.2	Technical Choice	15
4.2.1	General Tools	15
4.2.1.1	Back-End	15

4.2.1.2	Front-End	16
4.3	Physic architecture	17
4.4	Deployment diagram	17
5	Sprint 1 "Authentication"	19
5.1	Functional Specification:	19
5.1.1	Detailed use case diagram:	19
5.1.2	Conception:	20
5.1.2.1	Sequence diagram:	20
5.1.2.2	Activity Diagram:	23
5.1.2.3	Class Diagram:	24
5.1.3	Production	25
5.1.3.1	Login interface:	26
5.1.3.2	Create account interface:	27
5.1.3.3	Loading interface:	27
5.2	Conclusion:	27
6	Sprint 3 "Parent interface":	28
6.1	Functional Specification:	28
6.1.1	Sprint Backlog:	28
6.1.2	Sprint detailed use case diagram :	29
6.1.2.1	Add child use case diagram:	30
6.1.2.2	Profile use case diagram:	31
6.1.2.3	Track kids use case diagram:	32
6.1.3	Conception:	33
6.1.3.1	Sequence diagram:	33
6.1.3.2	Activity Diagram:	37
6.1.3.3	Class diagram:	39
6.2	Production:	41
6.2.1	Parent home interface:	41
6.2.2	Profile interface:	42
6.2.3	Chat interface:	43
6.3	Conclusion:	43
7	Sprint3: Child interface:	45
7.1	Functional Specification:	45
7.1.1	Sprint Backlog:	45
7.1.2	Sprint detailed use case diagram :	46
7.1.2.1	communication use case diagram:	46
7.1.3	Conception:	48
7.1.3.1	Sequence diagram:	48
7.1.3.2	Class diagram:	51
7.2	Production:	53
7.2.1	home interface:	53
7.2.2	Profile interface:	53

7.2.3	Chat interface:	54
7.3	Conclusion:	54

List of Figures

2.1	SCRUM life cycle	5
3.1	App architecture	9
3.2	Global use case diagram	10
3.3	Class diagram	11
4.1	Spring boot	16
4.2	Android	16
4.3	3-tiers architecture	17
4.4	Deployment diagram	18
5.1	detailed use case diagram	19
5.2	login sequence diagram	21
5.3	create account sequence diagram	22
5.4	login activity diagram	23
5.5	create account activity diagram	24
5.6	sprint 1 class diagram	25
5.7	login interface when the user choose to login as parent	26
5.8	login interface when the user choose to login as child	26
5.9	create account interface	27
6.1	Sprint 2 detailed use case diagram	29
6.2	add child use case diagram	30
6.3	Profile use case diagram	31
6.4	Track use case diagram	32
6.5	add child sequence diagram	33
6.6	profile sequence diagram	34
6.7	Call sequence diagram	35
6.8	Chat sequence diagram	36
6.9	Localization sequence diagram	37
6.10	Add child activity diagram	38
6.11	Sprint 2 class diagram	40
6.12	Parent home interface default	41
6.13	Parent home interface when newly added child is selected	41
6.14	Parent home interface when parent call a selected child	42
6.15	Parent profile interface	42

6.16	selected child profile interface	43
6.17	chat interface	43
7.1	Sprint 3 detailed use case diagram	46
7.2	communication use case diagram	47
7.3	profile sequence diagram	48
7.4	Call sequence diagram	49
7.5	Chat sequence diagram	50
7.6	Urgent help sequence diagram	50
7.7	localization sequence diagram	51
7.8	Sprint 3 class diagram	52
7.9	home interface	53
7.10	profile interface	53
7.11	chat interface	54

List of Tables

3.1	Product backlog	9
3.2	User Story partition by sprint	10
5.1	Text description of sprint 1 use cases	20
6.1	sprint 2 sprint backlog	28
6.2	Text description of sprint 2 detailed use cases	29
6.3	Text description of add child use cases	30
6.4	Text description of Profile use cases	31
6.5	Text description of Track kids use cases	32
7.1	sprint 3 sprint backlog	45
7.2	Text description of sprint 3 use cases	46
7.3	Text description of Communication use cases	47

Chapter 1 General introduction:

Since humanities population increasing in times so the crimes rates. For that kids safety are not easy to accomplish especially while their parent cannot be with them all time due to work or whatever reason can be. For that we thought of using the technologies progression to find a solution to help parents maintain their kids safety, which is an android application to localize kids.

- The first chapter “Project studies” will be an introductory chapter that will present the organization as well as the general framework of the work and the methodology adopted.
- The second chapter “Analysis and specification of need” where we will identify the actors and the needs, and specify the project backlog, application architecture and general conception.
- The third chapter “Sprint 0” for the working environment
- Chapter four, five and six present the “sprints” of our project, in which we return with more details to the design highlighting the functionalities offered by our solution, as well as a presentation of the stage of the realization and of the implementation of our solution.

Chapter 2 Project studies

Introduction

After presenting the general framework and determining the overall orientations of the subject. This chapter allows us to identify all the functionalities of our future system for each type of user, and this by identifying the functional needs and apprehending the list of requirements translated by the non-functional needs. This will be done by identifying the actors and defining all the needs which will be modeled by the general use case diagram.

2.1 Company presentation:



Mobelite is an agency specializing in mobile strategy consulting, design and development of mobile applications and websites and mobile marketing. Mobelite has a team of experts in the creation of mobile applications on the most popular platforms and web applications. Mobelite has sales and marketing teams in Paris (France), and development teams in Tunis and Monastir (Tunisia).

2.2 Study of the existing

With the time and technology progressing too fast it has to be an easy and quick for parent's to look out for their children to help them get throw many difficulty which is:

- Throw the difficulty of life parents can either forget or not have time to check their children.
- In a large world were is not safe for children they can get into a dangerous place without them or their parents notice.
- It may be difficult to communicate vocally and throw texting in a fast way
- it may be hard for human mind to memories all insecure places to adapt to it.

2.3 Problematical:

In order to remedy all its problems, we have assigned to our study the following objectives:

- Ease of research and access to information.
- Storage of information on computer support which will ensure their security.
- Automate tasks that are processed manually.
- Propose a good codification.
- Ensure a scalable and maintained application

2.4 Solution:

2.4.1 Existing solution:

The Highest rated app google play store is:

- Life360:Find familyAndFriends.
- KidsControl, Family GPS locator.
- Find My Kids: Parental Control

While some not free,And other has some feature make parent so obsessive about their kids.

2.4.2 Proposed solution:

After an extreme and detailed analysis of the client's needs and expectations.An android application has been proposed the proposed solution consists of two related, yet complementary parts:

- The first part is represented by a backend layer containing all the necessary endpoints and services responsible for all types of BD access, data transfer requests, upload download images, ...etc.
- - the second part is represented by a front-end layer that contains all the necessary interfaces for the user/parent or child to interact with. In this particular layer all the proposed chapters for the tracking process, communication, managing kids for parent user, and much more will be implemented.

2.5 Specification of requirement

The requirement specification step establishes a first description of the application. It is based on the result of the needs analysis and its feasibility to produce a presentation of what the application has to offer. To highlight our application, we must specify a list of requirements to meet the needs of users as well as ensure the proper functioning of our system:

- **Authentication:** The user need to be authenticated to access the application. So it can identifies parents and children.
- **Tracking:** Parent can track their children and customize the location for them.
- **Communication:** Parent and children's can communicate throw the application in a private space.
- **Tracking history:** Parent can see their children's tracking history.

2.6 Project management methodology

The choice between one development process model and another depends on the nature of the project and its size. When it comes to a project where the data is not gathered from the start, where the needs are incomplete or even unclear, it is recommended to move towards an iterative method or prototype oriented. Among the iterative methods, we can cite the widely used AGILE methods of our days around the world. An AGILE method is carried out in a collaborative spirit and adapts to incremental approaches. It generates high quality products while taking into account changing customer needs. It also enables continuous quality management and the detection of problems as soon as possible, thus allowing corrective actions to be taken without too many penalties in costs and delays. There are several AGILE methods and it is not to choose the best method among those existing. For our project, we have oriented towards an AGILE type method and more particularly SCRUM.

2.6.1 Presentation of Scrum

The principle of the SCRUM methodology is to develop software incrementally by maintaining a completely transparent list of requests for changes or corrections to to implement. With very frequent deliveries, the customer receives functional software each time iteration. The more the project progresses, the more complete the software is and always has more and more features .

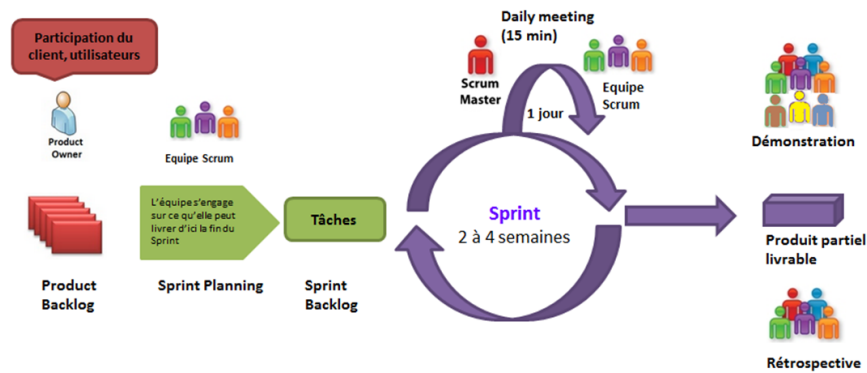


Figure 2.1: SCRUM life cycle

2.6.2 Roles in SCRUM

The SCRUM methodology involves three main roles which are:

2.6.2.0.1 Product owner: in most projects, the Product owner is the responsible for the client project team. He is the one who will define and prioritize the list of features product and choose the date and content of each sprint based on the values (charges) communicated to him by the team.

2.6.2.0.2 Scrum master: real facilitator on the project, he ensures that everyone can work to their full potential by removing obstacles and protecting the team from disruption exterior.

2.6.2.0.3 Team: the team organizes itself and remains unchanged for the duration of a sprint. She must do everything to deliver the product.

The life of a Scrum project is punctuated by a set of clearly defined and strictly limited in time (timeboxing):

1. **Sprint**(Sprint = iteration): Timeboxing is used to define the length of the Sprint. The Sprint is a one-month time box in which the scrum team will deliver the goals of the Sprint.
2. **Sprint planning:** : During this meeting, the development team selects the priority elements of the "Product Backlog" that she thinks she can carry out during the sprint (in agreement with the "Product Owner").
3. **The daily scrum:** The Daily Scrum is a 15-minute meeting. for each 24-hour period (also called a "stand up meeting") that helps the team to synchronize activities and make visible any obstacle to the achievement of the objective of the sprint. Each mainly answers 3 questions: "What have I finished since the last melee? What will I be done by the next scrum? What obstacles are holding me back? ".

4. **Sprint review:** During this meeting which takes place at the end of the sprint, the development team presents the features completed during the sprint and collects feedback from the Product Owner and end users. This is also the time to anticipate the scope of the next sprints and adjust the release planning as needed (number of sprints remaining).
5. **Sprint retrospective:** : The retrospective that has generally held after the sprint review is the opportunity to improve (productivity, quality, efficiency, conditions work, etc.) in the light of "experience" on the past sprint (principle of continuous improvement).

Conclusion

In this chapter, we started by presenting the general framework of the project where we introduced the host company. Then, we identified the problem through the study of the existing. Subsequently, we proposed the solution by following the existing tools on the market. Finally, we have chosen the methodology that will be used throughout the project, and by the then we will move towards the analysis and specification of the needs

Chapter 3 Analysis and specification of need

Introduction

This chapter presents the needs study which constitutes an analysis phase of the project. We will first present the main players and their needs. Then we will describe the Product Backlog, as well the planned sprints. Finally, we close the global class and use case diagrams.

3.1 Analysis of need

We present in the following the users of our application as well as the functional and non-functional needs.

3.1.1 Identification of actors

The analysis of an application begins with the determination of its different actors. A study of the interaction of the system with its external environment made it possible to identify two main players:

3.1.1.0.1 Parent: it the user how get most of the application functionality like add children to track

3.1.1.0.2 Child: it's the user how the app is going to keep track of. But if the app didn't found him (there is no parent for him in the database) he won't get access to the app

3.1.2 Functional need

This step consists of answering the question "what is our system for?" ". We group these needs into the following points:

- Authentication for parents or children's.
- Allow Parent's to track their children.
- Allow parents and children's to communicate with each other.

- Provide the application user information of:
 - Their application profiles.
 - The children’s application profile to their parent.
- Allow parent to add their children.
- Allow parent to adjust location security level.
- Notify parent when their children’s in non safe location.

3.1.3 Non functional need

Non-functional requirements are requirements that are visible to the user, but that are not directly related to the behavior of the system. The needs of our system are described as follows:

- **Extensibility:** The architecture of the application will allow the evolution and maintenance (addition or deletion or update) at the level of its various modules in a flexible manner.
- **Reliability:** The system must ensure that the application server is available at all times without errors and that no failure could occur.
- **The rapidity:** The system must ensure speed of execution.
- **Performance:** it is a question of optimizing the time of loading of the data from two different environments as well as by the use of the best practices of the development.

3.2 Product backlog

The Product Backlog is a prioritized list of customer needs and requirements. Items in the Product Backlog, also known as User Stories, are worded in one or two sentences that clearly and precisely describe the functionality the customer wants, typically written in the following form, "As X, I want Y, in order to Z ". The Product Backlog shown in Table 3.1 includes the following fields:

- **ID:** This is a unique, auto-incrementing number for each user story.
- **User story:** It is a sentence describing the functionality desired by the customer.
- **Priority:** It is the business value that drives the prioritization of the development of user stories according to the expectations and needs of the customer, ranging from 0 to 100.

ID	User story	Priority
1	As user i need to identify my self as parent or child	100
2	As parent i can create an account or sign in if i have one	100
3	As child i can sign in into an existing account	80
4	As an authenticated user i can chat an call other user	50
5	As parent i can track my children and disable it	100
6	As parent i can add my children to the app for tracking	80
7	As child i can ask for emergency help	90
8	As parent i can check my child navigation history and configure location security for them	70
9	As parent i get to be notified when my children's location not safe	100

Table 3.1: Product backlog

3.3 Application architecture

To develop this application we need a server to communicate between database and our app like figure 3.1 describe

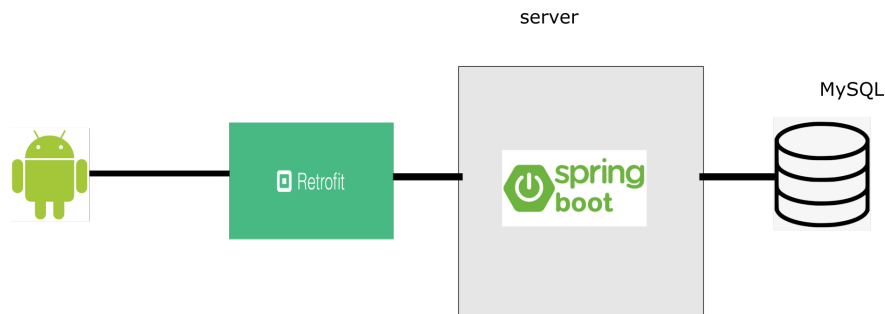


Figure 3.1: App architecture

This architecture describes well how our application work:

1. Store the data in MySQL database.
2. Restful web service using Spring boot, which follow the MVC pattern .
3. Restful API (Retrofit) to extract data from a the server.
4. Android application using android framework which is follow the MVVM pattern.

3.4 Sprint planification

The "User stories" previously defined in the Product Backlog are sorted in order of priorities and business values. The goal is to implement what has the most value first. The work will be planned according to the sprints that we have defined and each one lasts one to four weeks.

Sprint	Sprint name	Period
Sprint 1	Authentication	1 mounth
Sprint 2	Parent interface	1 mouth
Sprint 3	Child interface	1 mouth
Sprint 4	security and notification	1 mouth

Table 3.2: User Story partition by sprint

3.5 Modeling of functional needs

The functionalities mentioned above will be presented in the following part by a global use case diagram and a global class diagram. Recognized as being the industry standard par excellence for object modeling, we opted for the UML modeling language. This illustration will make it possible to better structure the interactions between the actors and the system.

3.5.1 Global use case Diagram

In this section, we present the needs of our system in a formal way using the UML modeling language use case diagram. It represents the structure of the major functionalities required by the users of the system. Figure 3.2 shows the overall use case diagram of our application

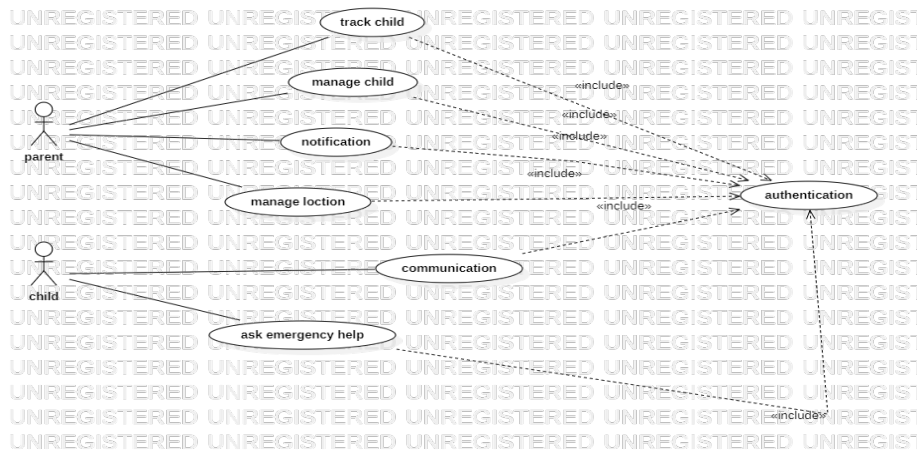


Figure 3.2: Global use case diagram

3.5.2 Class diagram

The class diagram is the focal point in object-oriented development. It represents the structure of an object-oriented code or, at a greater level of detail, the development language modules .

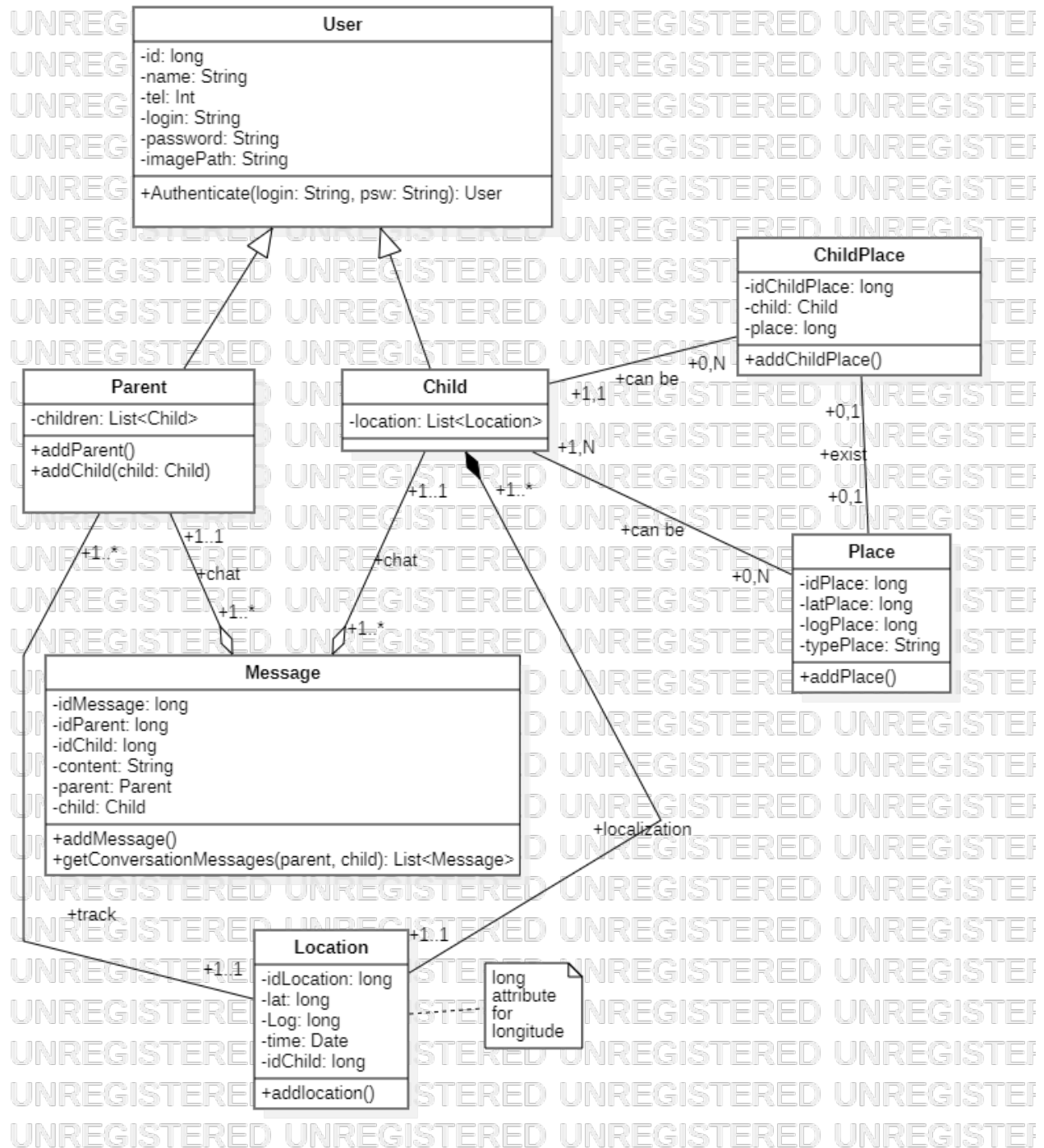


Figure 3.3: Class diagram

Conclusion

In conclusion, we can say that the needs analysis provides a clearer view of the subject and a deeper understanding of the tasks at hand. It also leads to predicting problems and finding and seeking solutions to circumvent them. Therefore, we must specify and seek the possible means to implement these solutions in order to achieve the objectives of the work. In this chapter, we have highlighted the needs of our application. Moreover, this analysis allowed us to better clarify the ideas concerning the functionalities to be developed. In addition, we will identify the technical study of the project in the next chapter.

Chapter 4 Sprint0”Technical study”

Introduction

In this chapter, we present the first phase of Scrum which is Sprint 0. For this reason, we describe the work environment and the technical choices that allowed us to implement the design discussed for this project.

4.1 Development Environment and technical choice

4.1.1 Hardware environment

During the development of the project, we used a AOpen computer which had the following characteristics:

- **Aopen:**
 - Operating system: windows 10/64bits
 - RAM: 8 GO
 - CPU: Intel(R) Core(TM) i3-3220
 - GPU: NVIDIA gt610

4.1.2 Development environment

In this section, we specify the tools and technologies used for the design and realization of our project as follows:

4.1.2.1 Modeling tool

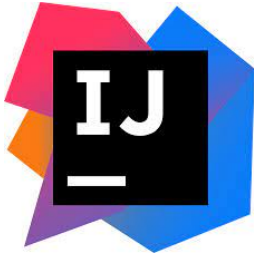


StartUML[1]: is an open-source Unified Modeling Language (UML) modeling software, dedicated to Windows platforms. Also, it is the standard for the exchange of XML-based UML metadata information.

4.1.2.2 Integrated development environment



Android studio[2]: 2 is a free, cross-platform (Windows, Mac, and Linux) code editor designed for mobile application development with Java and Kotlin.



IntelliJ[3]: What is IntelliJ used for? IntelliJ IDEA is an Integrated Development Environment (IDE) for JVM languages designed to maximize developer productivity.



MySQL[4]L: MySQL is a database management system (DBMS). Its role is to record data in an organized and consistent way to help you find it easily later. MySQL derives directly from SQL (Structured Query Language) which is a query language to databases using the relational model. MySQL, depending on the type of application, its license is free or proprietary. It is one of the most popular database management software used in the world, both by the general public (mainly web applications) and by professionals



Spring[5] : The Spring Framework is an application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform.



Texstudio[6]: TeXstudio is an integrated writing environment for creating LaTeX documents. By Therefore, TeXstudio has many features such as highlighting the syntax, the built-in viewer, reference checking and various wizards. TeXstudio is open source and is available for all major operating systems.



Postman[7]: is a tool for testing/calling a web API. Postman comes in two flavors: an in-browser app and a packaged app, both for Chrome. The features are similar but not strictly identical. Once Postman is installed, we have a complete graphical environment to manage all interactions with our Web APIs. Postman proposes the collections: It is thus possible to organize the requests in groups (the collections), themselves being able to benefit from directories to facilitate navigation in the event of large number of requests.

4.2 Technical Choice

We present, in this part, the global technologies with which we carry out our project

4.2.1 General Tools

4.2.1.1 Back-End

Java Spring Framework is a popular, open source, enterprise-level framework for creating standalone, production-grade applications that run on the Java Virtual Machine (JVM).

Java Spring Boot is a tool that makes developing web application and microservices with Spring Framework faster and easier through three core capabilities:

1. Autoconfiguration.
2. An opinionated approach to configuration.
3. The ability to create standalone applications.

These features work together to provide you with a tool that allows you to set up a Spring-based application with minimal configuration and setup.

Spring Framework offers a dependency injection feature that lets objects define their own dependencies that the Spring container later injects into them. This enables developers to create modular applications consisting of loosely coupled components that are ideal for microservices and distributed network applications.

The biggest advantages of using Spring Boot versus Spring Framework alone are ease of use and faster development. In theory, this comes at the expense of the greater flexibility you get from working directly with Spring Framework.

But, in practice, unless you need or want to implement a very unique configuration, using Spring Boot is worth the trade off. You still are able to use Spring Framework's very popular annotation system that lets you easily inject extra dependencies into your application. And, you still get access to all Spring Framework features, including easy

event handling, validation, data binding, type conversion, and built-in security and testing capabilities. Bottom line, if your project's scope is covered by even just one Spring Starter, Spring Boot can significantly streamline development.



Figure 4.1: Spring boot

4.2.1.2 Front-End

is the process by which applications are created for devices running the Android operating system. Google states that "Android apps can be written using Kotlin, Java, and C++ languages" using the Android software development kit (SDK), while using other languages is also possible. All non-Java virtual machine (JVM) languages, such as Go, JavaScript, C, C++ or assembly, need the help of JVM language code, that may be supplied by tools, likely with restricted API support. Some programming languages and tools allow cross-platform app support (i.e. for both Android and iOS). Third party tools, development environments, and language support have also continued to evolve and expand since the initial SDK was released in 2008

The Android software development kit (SDK) includes a comprehensive set of development tools. The Android SDK Platform Tools are a separately downloadable subset of the full SDK, consisting of command-line tools such as adb and fastboot. The Android Debug Bridge (ADB) is a tool to run commands on a connected Android device. Fastboot is a protocol used for flashing filesystems. Code written in C/C++ can be compiled to ARM, or x86 native code (or their 64-bit variants) using the Android Native Development Kit (NDK).



Figure 4.2: Android

4.3 Physic architecture

A web application generally follows the three-tier architecture model. Our application is based on the client/server concept, the different elements of which are considered as distinct elements hence the use of this architecture.

A distributed client/server architecture is generally an architecture shared between:

- **Client:** the one who asks for the resources.
- **Application Server:** which is a middleware responsible for providing the resource but doing call to another server.
- **Data source:** typically a database server, providing a service to the first server.

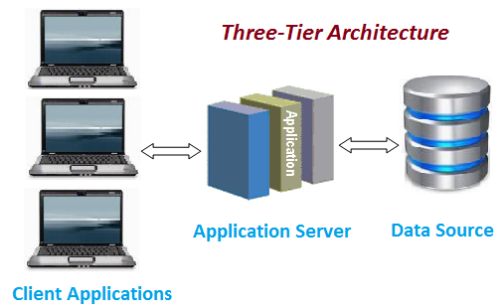


Figure 4.3: 3-tiers architecture

The 3-tier approach, as shown in Figure 4.3, has several advantages. Among its advantages:

- A separation of data resulting in a design that is clearer and more efficient.
- A time saving for maintenance and evolution of the app.
- Greater flexibility to organize the development of the app between different developers.
- Independence between the three levels.
- More flexibility in resource allocation and client-to-server requests.

4.4 Deployment diagram

Deployment diagrams are even closer to physical reality, since they identify the hardware elements (PC, Modem, Workstation, Server, etc.), their layout physical (connections) and the layout of executables (represented by components) on these material elements. Figure below shows the physical structure of our computer system and the distribution of software components on this system by this deployment diagram.

The deployment of the application follows the 3-tier architecture which consists of:

- **Client:** Means the android application.
- **Server:** it is an Tomcat server capable of interpreting requests and providing replies through port 80.
- **Database server:** he MySQL database management system that contains the tables of our app.

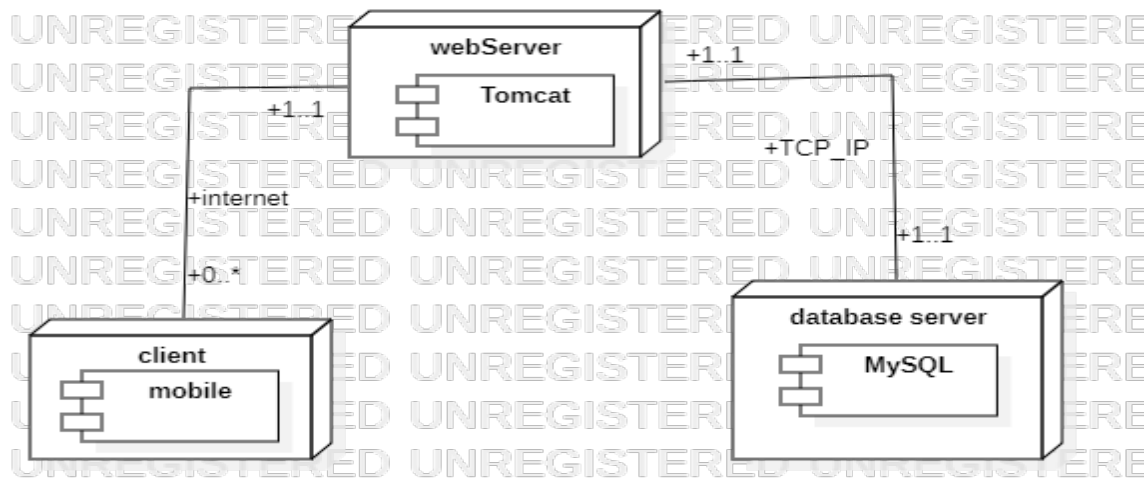


Figure 4.4: Deployment diagram

Conclusion:

In this chapter, we started with the first phase of Scrum where we presented the development environment and the different technical choices used as well as the technical architecture of our application. In the following chapters we will begin the analysis of our release

Chapter 5 Sprint 1 "Authentication"

Introduction

This chapter is the subject of a presentation of the first sprint of the project which is the Authentication. It is composed of Login and creating account sprint 1. The study of sprint covers the analysis, the conception, and realization.

5.1 Functional Specification:

5.1.1 Detailed use case diagram:

We then present the analysis phase of which we describe the use case diagram and the textual description. Figure 5.1 describes the detailed use case diagram of the sprint as well as its textual description in table 5.1.

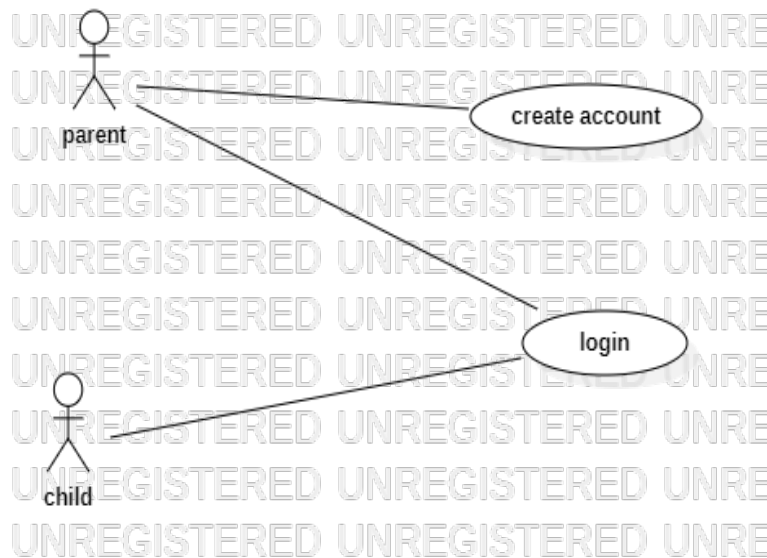


Figure 5.1: detailed use case diagram

Clause	Description
Identify	Build and manage variable of users.
Summary	user authenticate depend on it's type
Actors	Parent and children
Trigger context	Any user interested in the app functionality
Pre conditions	User need to identify himself as child or not
Description	Depending on whether user is child or not,he has the ability the ability to just log in or create account with it

Table 5.1: Text description of sprint 1 use cases

5.1.2 Conception:

In this part, we move on to the second activity in the product management sprint. variables which is the design where we present the diagrams of sequences, activities as well than the class diagram for this sprint

5.1.2.1 Sequence diagram:

Sequence diagrams document the interactions to be implemented between objects temporally to achieve an outcome, such as a use case. The diagram of sequence lists objects horizontally, and time vertically. It models the execution different messages depending on the time.

The diagram shown in Figure 5.2 is the sequencing of interactions between the user and the objects of the system responsible for authentication which offers the user an authentication form.

5.1.2.1.1 Login: As soon as the client validates after entering the email and password after choosing is he a parent or a child, then the app will verify if the user is authenticated or not with the server using retrofit service as a communication tool with the server. Means if it is successful (the user is authenticated) navigate to login page else display error message.

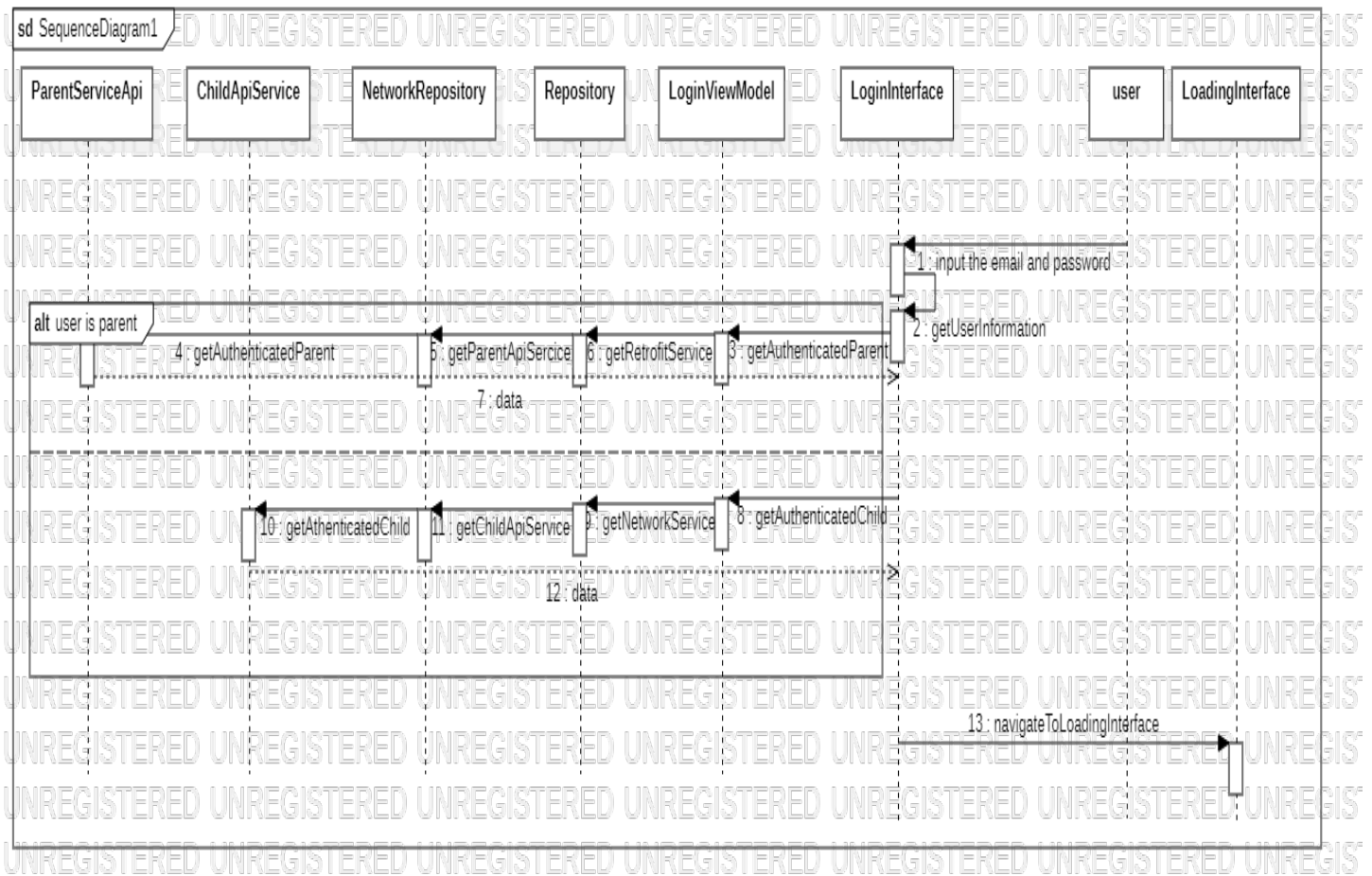


Figure 5.2: login sequence diagram

5.1.2.1.2 Create account: From login interface only parent will access create account interface by clicking on the create button the he input all the information needed with photo as his profile picture. Because child will created by his parent in from parent home interface. And the rest is same as the login as the figure 5.3 show

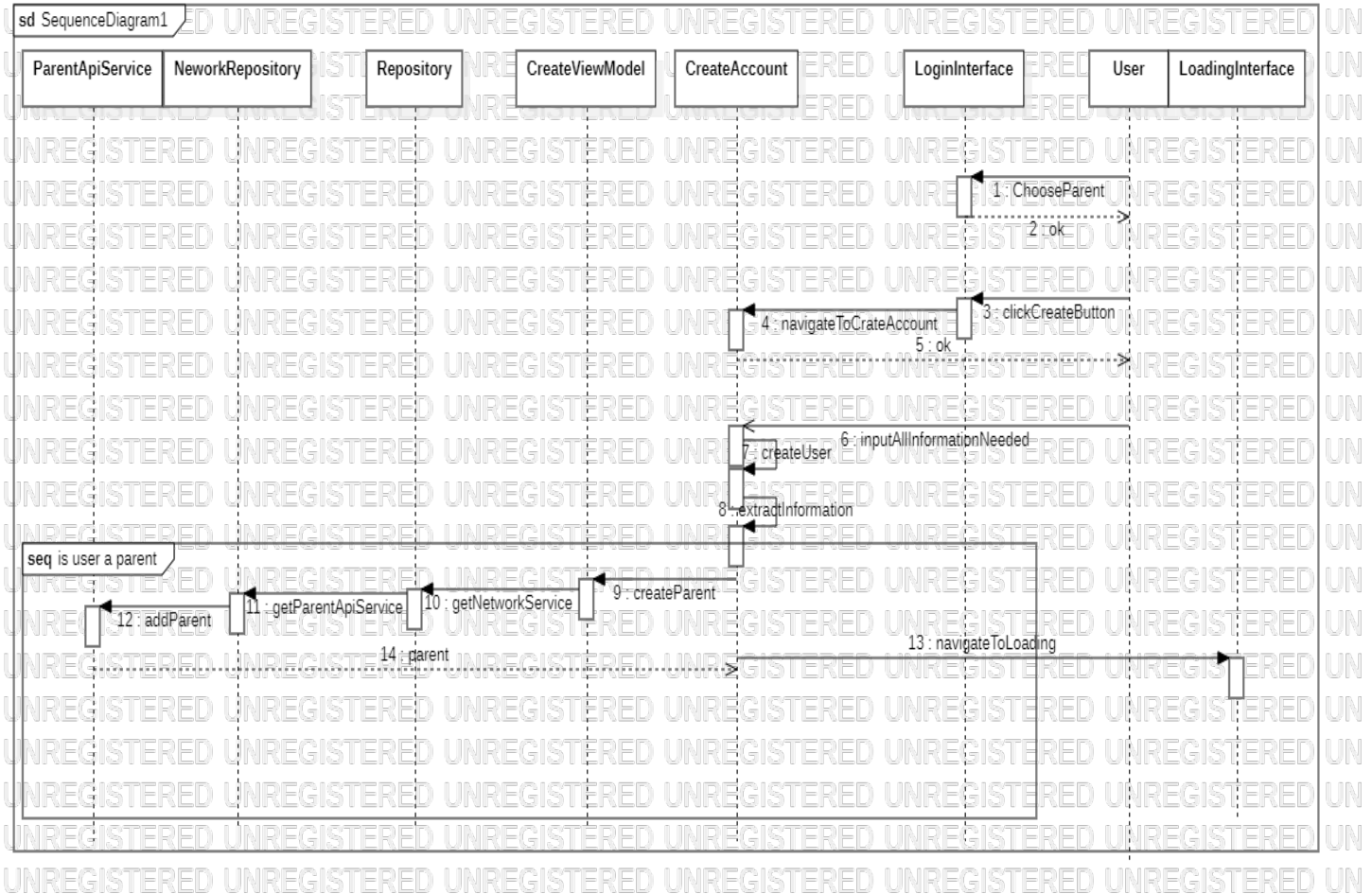


Figure 5.3: create account sequence diagram

5.1.2.2 Activity Diagram:

Activity diagrams allow you to focus on treatments. So, they are particularly suitable for modeling the routing of control flows and control flows. data. They thus make it possible to graphically represent the behavior of a method or the unfolding of a use case.

5.1.2.2.1 Login: Login consists of verifying the identity of an entity, in order to authorize its access to the system. It therefore makes it possible to validate the authenticity of the entity in question. Figure 5.4 describes the following activities:

- An authentication form is displayed for this user
- The user choose to login as parent or child
- The user enters their email and password
- The system checks the user's access credential in data base
- If he is authenticated the system will take him to loading page

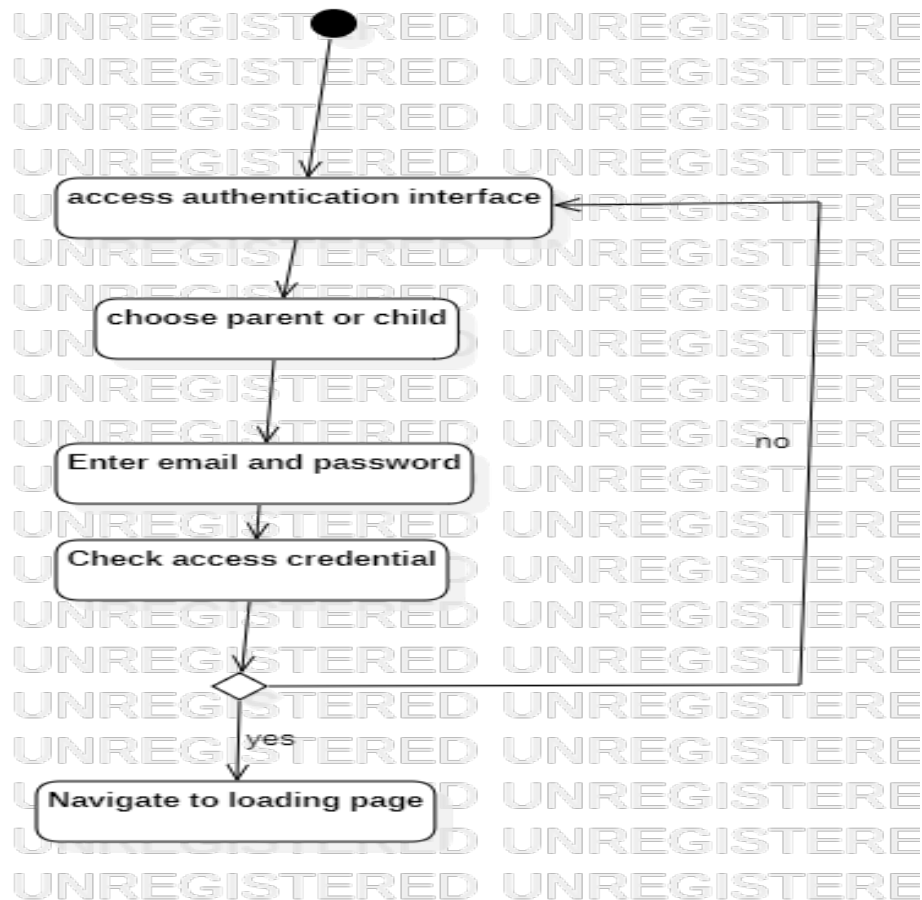


Figure 5.4: login activity diagram

5.1.2.2.2 create account: create consists of creating the identity of an entity, in order to authorize its access to the system. Figure 5.5 describes the following activities:

- If user choose parent the create button will be visible
- click on the create button
- An create account form is displayed
- The user enter their information
- The system will try to add him to the data base
- If the creation success he will go to the loading page

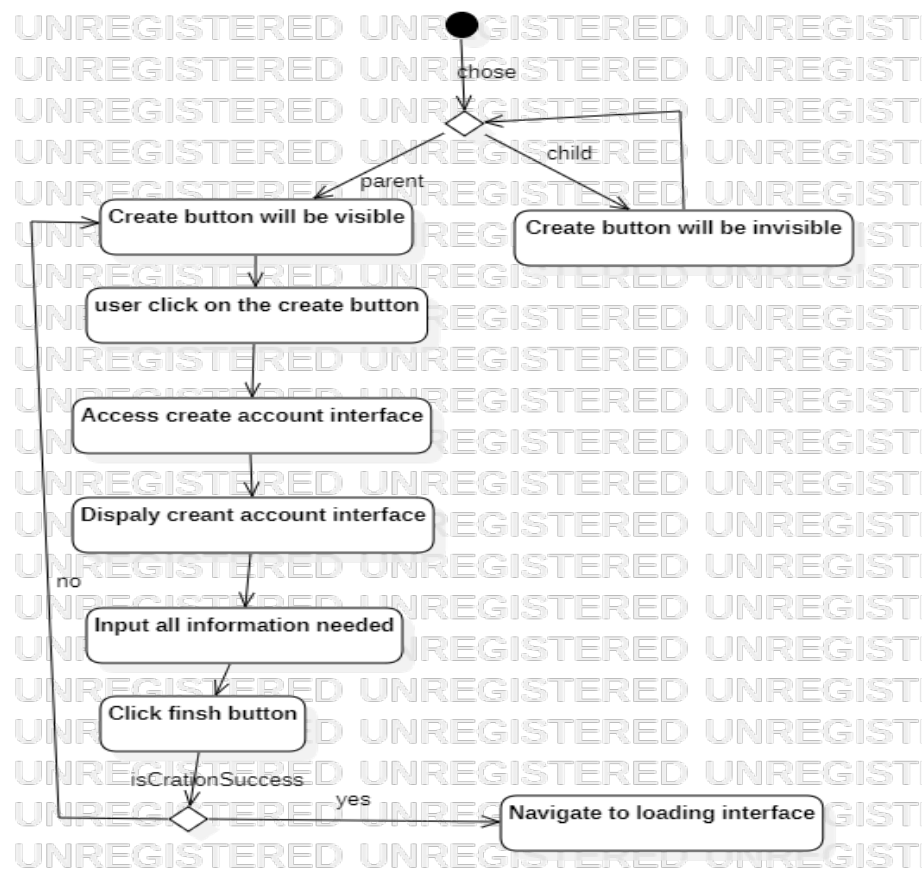


Figure 5.5: create account activity diagram

5.1.2.3 Class Diagram:

The class diagram makes it possible to describe the internal structure while showing the different classes, their attributes as well as the different structural relationships between these classes. It's about from a static view, because we do not take into account the time

factor in the behavior of system. The class diagram models the concepts of the application domain as well as the internal concepts created from scratch as part of the implementation of an application. Each Object-Oriented Programming language gives a specific way to implement the paradigm object (pointers or not, multiple inheritance or not, etc.), but the class diagram allows you to model the classes of the system and them relationships independently of a programming language particular.

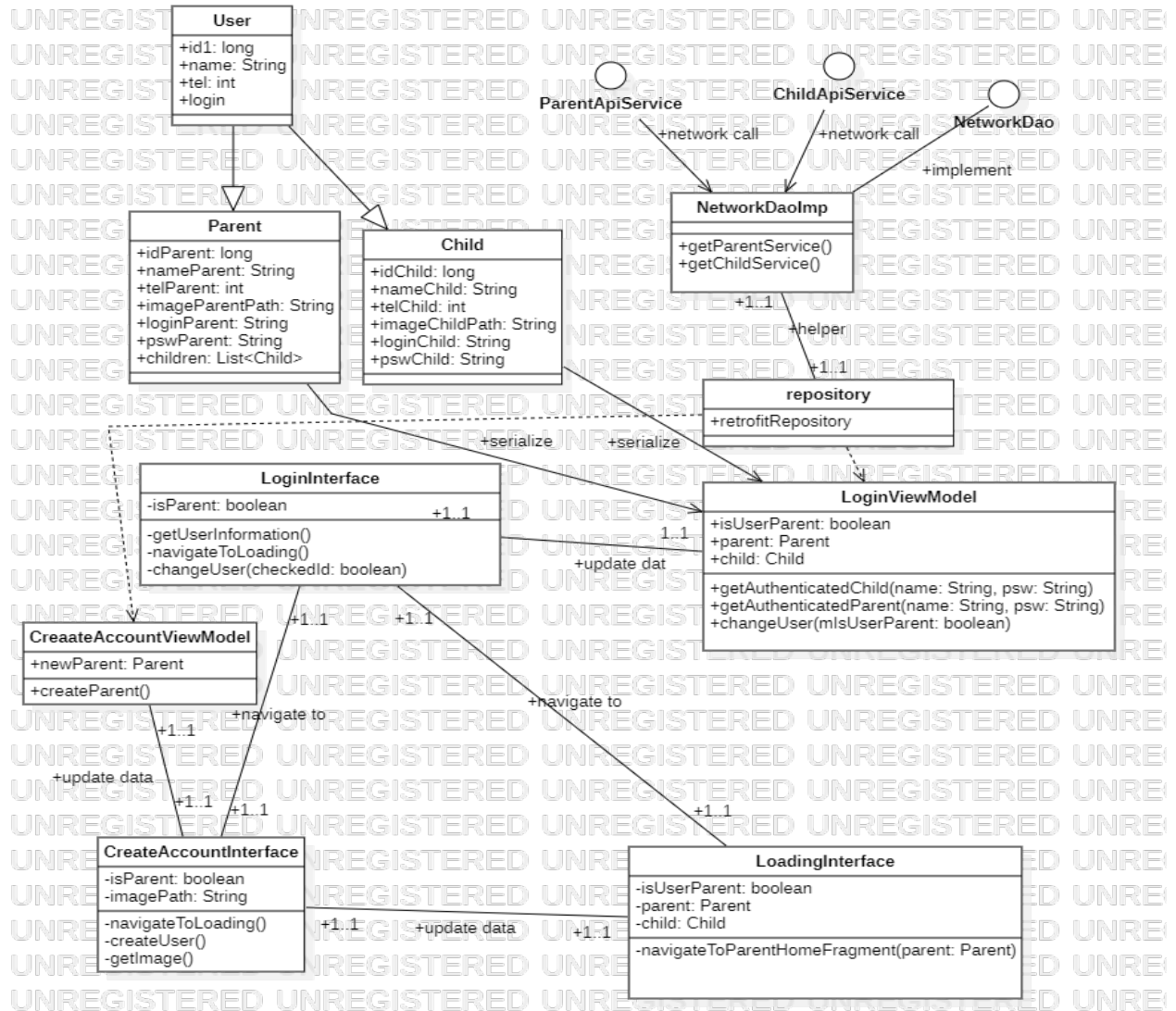


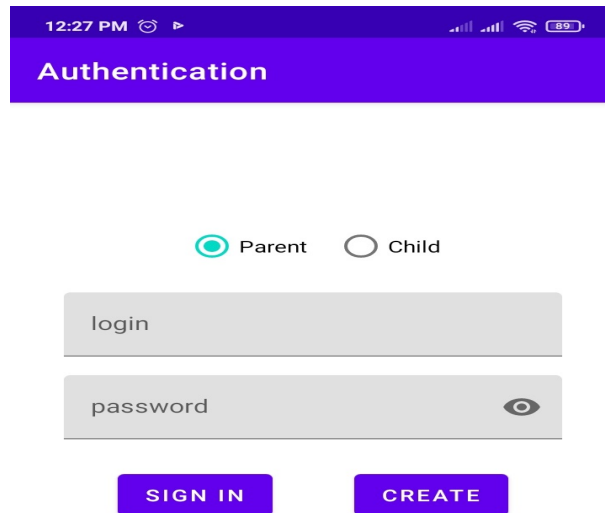
Figure 5.6: sprint 1 class diagram

5.1.3 Production

In order to show the results of this sprint, we will expose in this section the interfaces Man-Machine through the different screen prints produced.

5.1.3.1 Login interface:

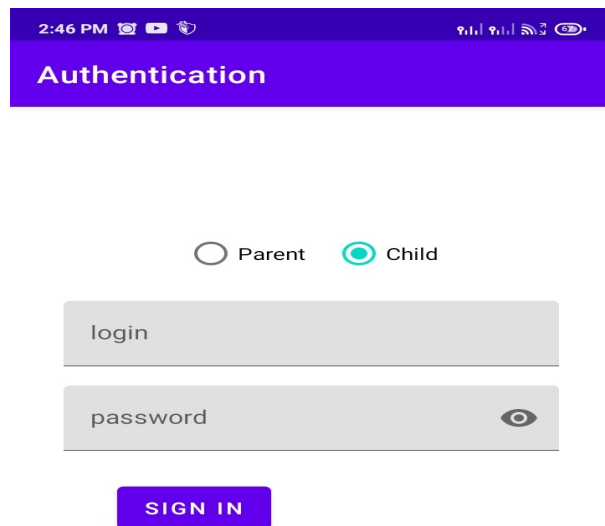
The figure 5.7 and 5.8 show the login interface, where the user decide to chose to login as parent or as child ,so the ui will change accordingly.



The image shows a mobile app interface for login. At the top, a status bar displays the time 12:27 PM, signal strength, and battery level at 89%. Below the status bar is a purple header with the word "Authentication" in white. Under the header, there are two radio buttons: "Parent" (selected, indicated by a green dot) and "Child". Below the radio buttons are two text input fields: "login" and "password". The "password" field has a green eye icon on the right side. At the bottom, there are two purple buttons: "SIGN IN" and "CREATE".

Figure 5.7: login interface when the user choose to login as parent

The figure 5.7 represent the ui when the user choose to login as parent



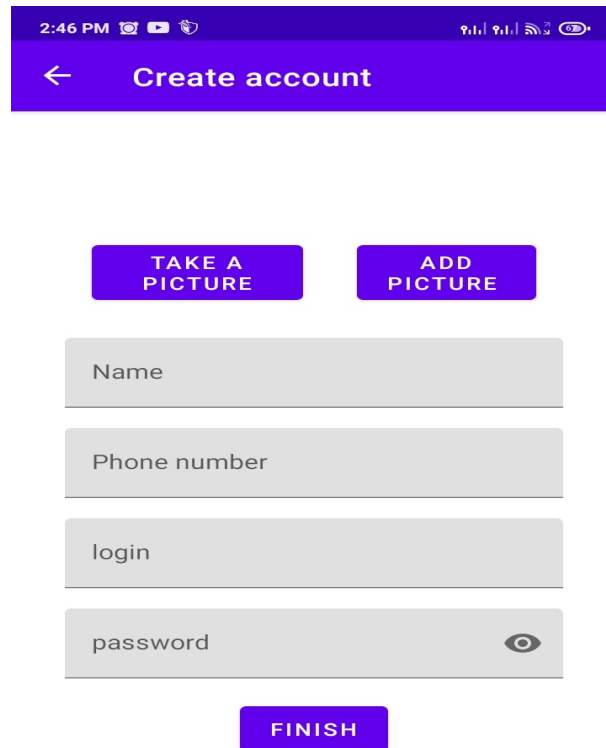
The image shows a mobile app interface for login. At the top, a status bar displays the time 2:46 PM, signal strength, and battery level at 89%. Below the status bar is a purple header with the word "Authentication" in white. Under the header, there are two radio buttons: "Parent" (unselected) and "Child" (selected, indicated by a green dot). Below the radio buttons are two text input fields: "login" and "password". The "password" field has a green eye icon on the right side. At the bottom, there is a single purple button: "SIGN IN".

Figure 5.8: login interface when the user choose to login as child

The figure 5.8 represent the ui when the user choose to login as child

5.1.3.2 Create account interface:

The figure 5.9 show the create account interface, this interface only accessible to parent users .Because only them can add their children as an app user and only they can add themselves.



2:46 PM

← Create account

TAKE A PICTURE

ADD PICTURE

Name

Phone number

login

password

FINISH

Figure 5.9: create account interface

5.1.3.3 Loading interface:

The figure 5.10 show the loading interface , in this interface the app will download all images related to the user

5.2 Conclusion:

In this chapter, we have described the steps carried out during the first sprint which was devoted to authentication. We started with a functional specification, then we have detailed the design by presenting the sequence, activity and class diagrams. Finally, we ended with the presentation of the realization via the Man-Machine interfaces. The next chapter is devoted to the second release dedicated to the Data manipulation.

Chapter 6 Sprint 3 "Parent interface":

Introduction: This chapter is the subject of a presentation of the second sprint of the project which is the realization of the app accessible to the parent users. the study of the analysis of the sprint, the conception, and realization.

6.1 Functional Specification:

6.1.1 Sprint Backlog:

After setting the goal of sprint, we define which "user stories" will be the subject of the Backlog for this sprint. We present through table 6.1 the different tasks of each user story.

User stories	priority
Tack kids location	100
Chat with the kids	80
Add kids	90
call the kids	70
see parent and kids profile	60
log out	50

Table 6.1: sprint 2 sprint backlog

6.1.2 Sprint detailed use case diagram :

We then present the analysis phase of which we describe the use case diagram and the textual description. Figure 6.1 describes the detailed use case diagram of the sprint as well as its textual description in table 6.2

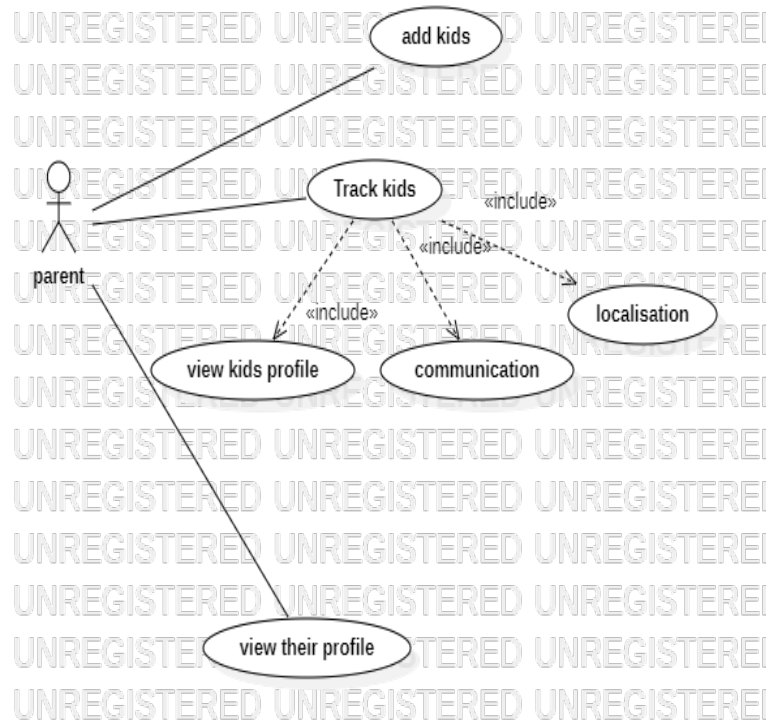


Figure 6.1: Sprint 2 detailed use case diagram

Clause	Description
Identify	parent functionality.
Summary	Parent like administrator for this app.
Actors	Parent
Trigger context	Authenticated parent
Pre conditions	parent related images downloaded
Description	When the parent is authenticated and his related images has been downloaded he can access the parent features of the app

Table 6.2: Text description of sprint 2 detailed use cases

6.1.2.1 Add child use case diagram:

Figure 6.2 illustrates the diagram detailing the use case for add child as well as its textual description presented in table 6.3.



Figure 6.2: add child use case diagram

Use case	add child
Actor	Parent.
Pre-condition	Parent access the Create account interface.
Post Condition	Child added to the data base and displayed in the parent home interface.
Nominal scenario	<ul style="list-style-type: none">• Get the parent input.• Check the validation of the input.• Send Post HTTP request to the data base.• The server check if there are any other child user with the same login,if not add him to the data base• Navigate back to the parent home interface.

Table 6.3: Text description of add child use cases

6.1.2.2 Profile use case diagram:

Figure 6.3 illustrates the diagram detailing the use case for profile as well as its textual description presented in table 6.4

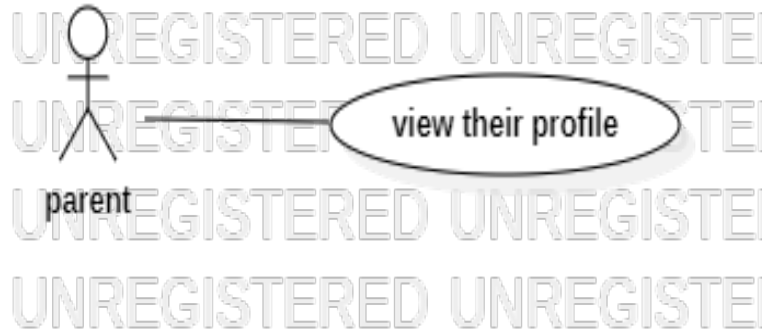


Figure 6.3: Profile use case diagram

Use case	Profile
Actor	Parent.
Pre-condition	Parent click the profile icon in the top bar.
Post Condition	Parent Profile displayed.
Nominal scenario	<ul style="list-style-type: none">• Process the parent data.• get his image URI for the android file system.• display his information in the ui.

Table 6.4: Text description of Profile use cases

6.1.2.3 Track kids use case diagram:

Figure 6.4 illustrates the diagram detailing the use case for profile as well as its textual description presented in table 6.5.

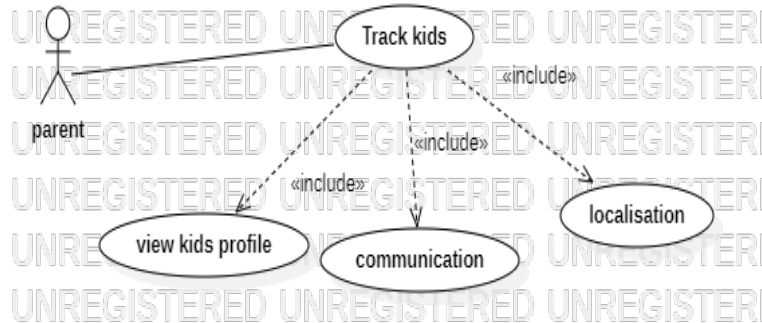


Figure 6.4: Track use case diagram

Use case	Track kids
Actor	Parent.
Pre-condition	Parent access the parent home inter-face.
Post Condition	Parent access all the feature the app offer related to his kids.
Nominal scenario	<ul style="list-style-type: none">• Parent chose a child from list of kids.• Parent can see his child loca-tion immediately if there is any.• Parent can chat with the se-lected child and call him .• Parent can view the selected child profile.

Table 6.5: Text description of Track kids use cases

6.1.3 Conception:

In this part, we move on to the second activity in the product management sprint. variables which is the design where we present the diagrams of sequences, activities as well than the class diagram for this sprint.

6.1.3.1 Sequence diagram:

Sequence diagrams document the interactions to be implemented between objects temporally to achieve an outcome, such as a use case. The diagram of sequence lists objects horizontally, and time vertically. It models the execution different messages depending on the time.

6.1.3.1.1 Add child sequence diagram: The figure 6.5 show how the child can be added. When parent want to add his child to befit from the app functionality on his newly added child he need to click on add child icon on the app bar,which will take him to the create account interface so he can fill the format with his child information an click on the finish button so the system download the image of his new child the take him back to the parent home interface.

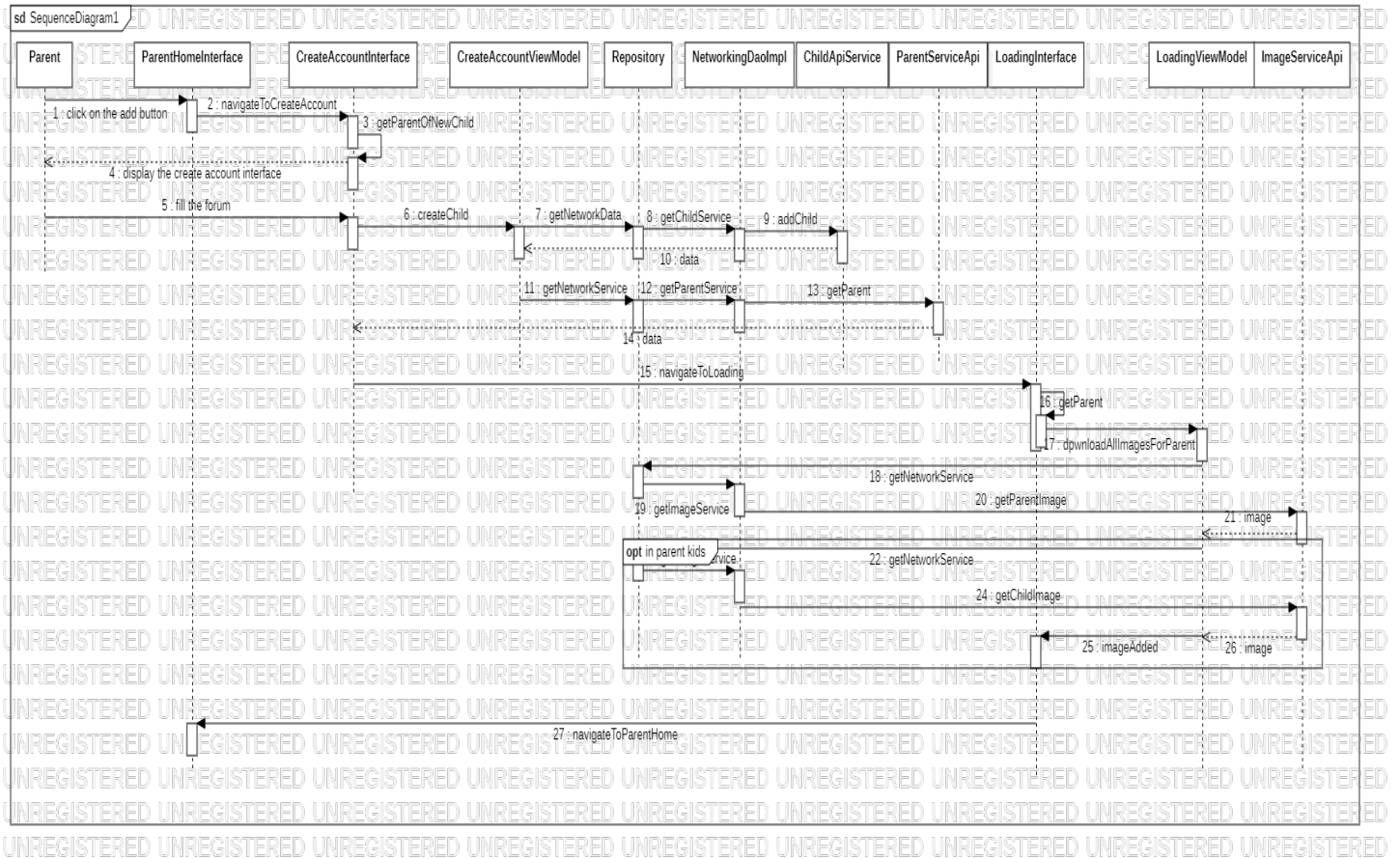


Figure 6.5: add child sequence diagram

6.1.3.1.2 Profile Sequence diagram: The figure 6.6 show how profile work. After parent click on the profile icon in top bar ,he will be considered as the user to be display its profile and to be recognized to be as parent.so his profile will be display the he will navigate to the profile interface who will look for his uri image in the android file system the show his profile

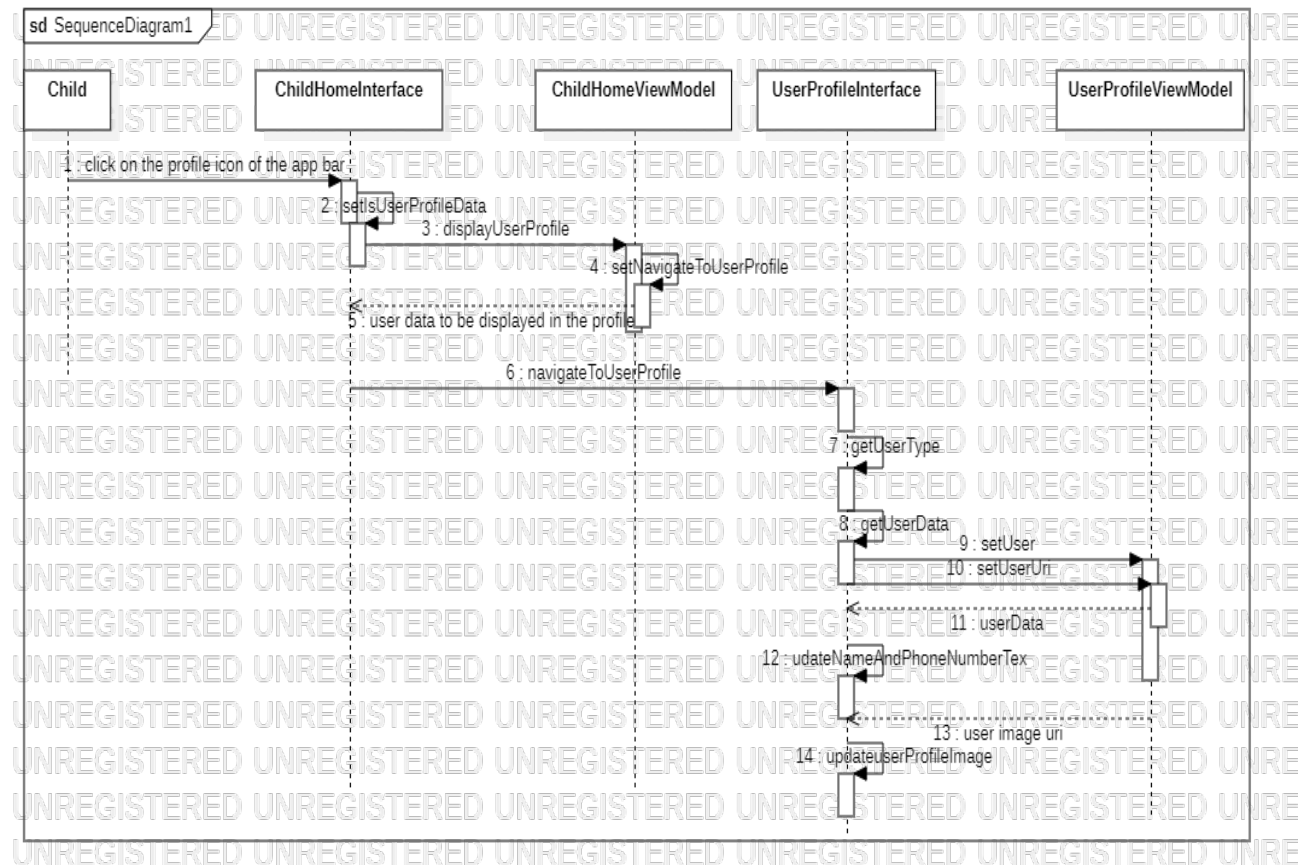


Figure 6.6: profile sequence diagram

6.1.3.1.3 Track kids(call): The figure 6.7 show how parent can call their child. first he need to grant permission or the app wont be able to make the call,if he grant the permission then he wont be asked for permission again unless he go to the android setting for the app and disable the permission.

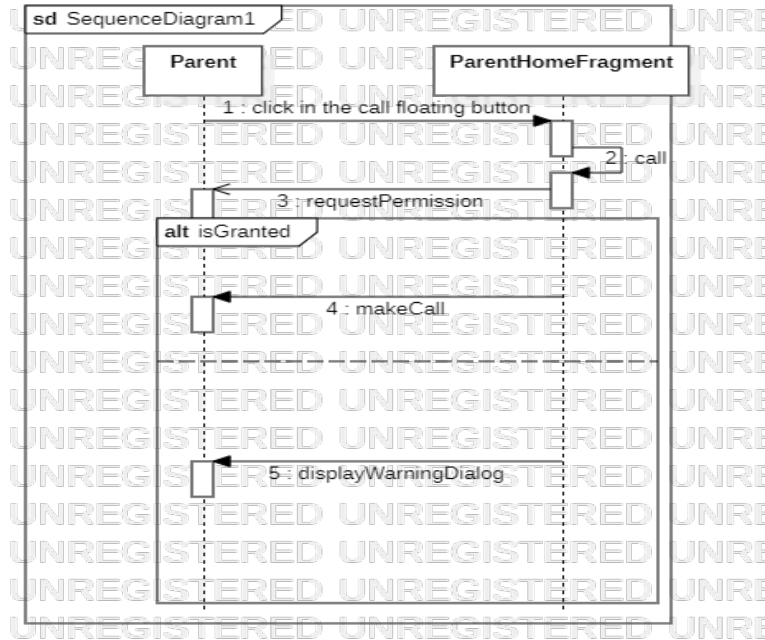


Figure 6.7: Call sequence diagram

6.1.3.1.4 Track kids(chat): The figure 6.8 show how parent can chat with their child. After selecting a child parent click the chat button to start chatting.

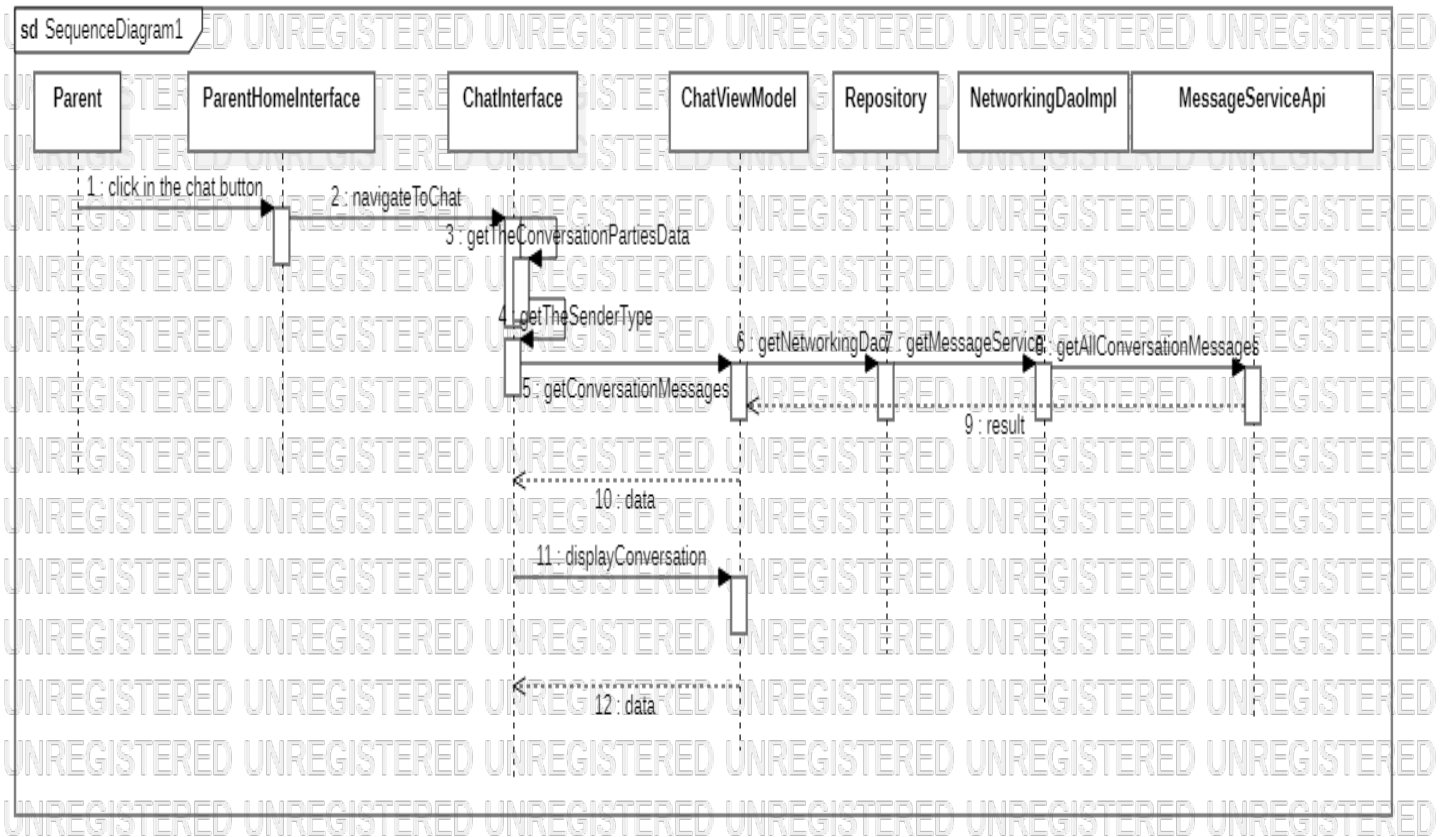


Figure 6.8: Chat sequence diagram

6.1.3.1.5 Track kids(localization): The figure 6.9 show how parent can track their kids location

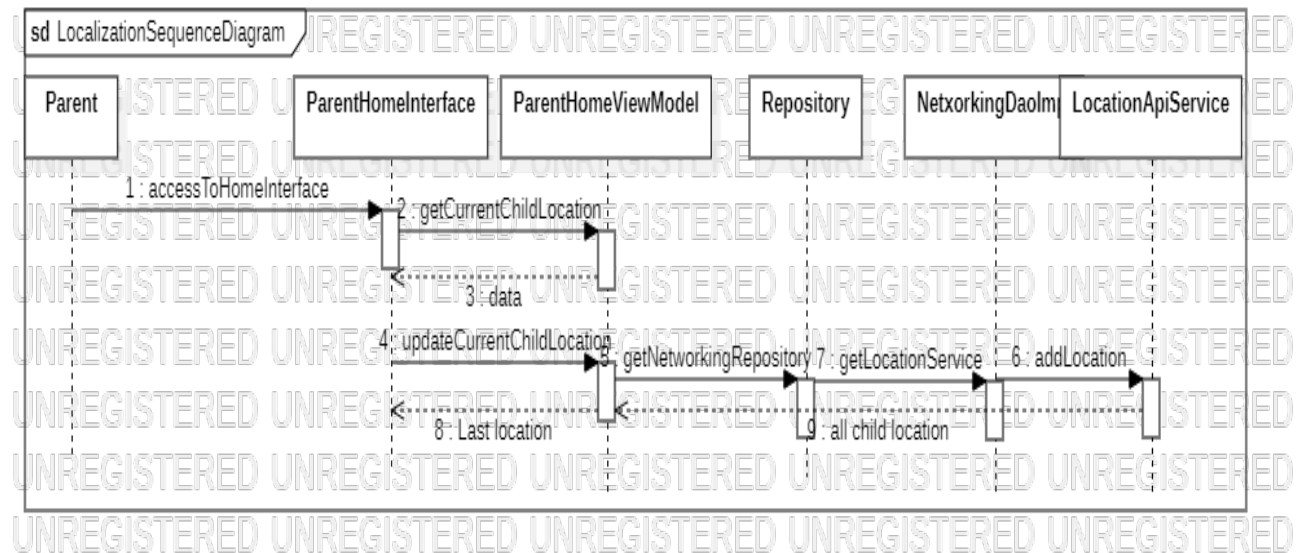


Figure 6.9: Localization sequence diagram

6.1.3.2 Activity Diagram:

Activity diagrams allow you to focus on treatments. So, they are particularly suitable for modeling the routing of control flows and control flows. data. They thus make it possible to graphically represent the behavior of a method or the unfolding of a use case.

6.1.3.2.1 Add child Activity diagram: Add child consist the ability for a father to control his child information for the app and him alone has the ability to add one. Figure 6.9 describes the following activities:

- The user request to add a child clicking n the add child button.
- Access to the create account interface.
- The create account displayed.
- The parent fill the forum.
- System check the input validity .
- If the input invalid it will display an error message.
- Else navigate to loading interface.
- Display the loading interface.

- Download the images.
- navigate to parent home interface.

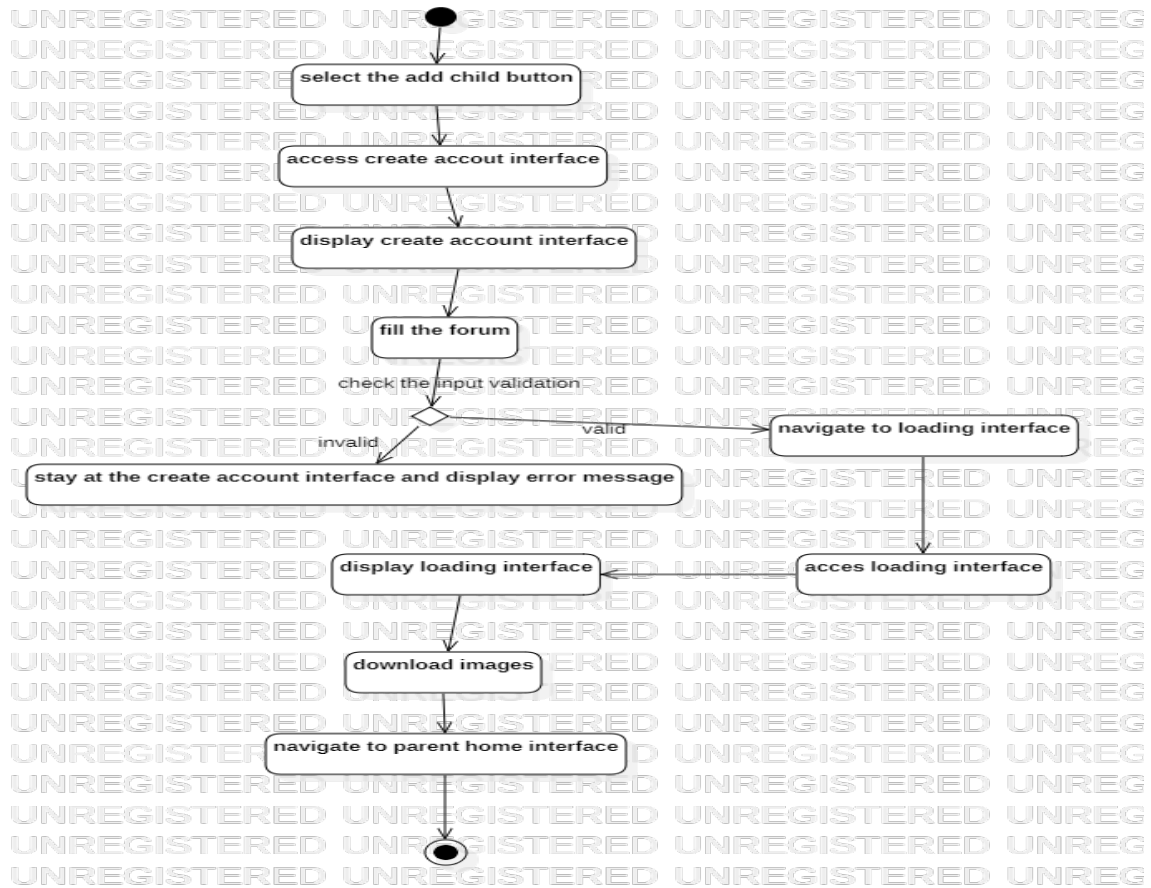


Figure 6.10: Add child activity diagram

6.1.3.3 Class diagram:

The class diagram makes it possible to describe the internal structure while showing the different classes, their attributes as well as the different structural relationships between these classes. It's about from a static view, because we do not take into account the time factor in the behavior of system. The class diagram models the concepts of the application domain as well as the internal concepts created from scratch as part of the implementation of an application. Each Object-Oriented Programming language gives a specific way to implement the paradigm object (pointers or not, multiple inheritance or not, etc.), but the class diagram allows you to model the classes of the system and them relationships independently of a programming language particular. Figure 5.10 describes the class diagram we used to develop the sprint.

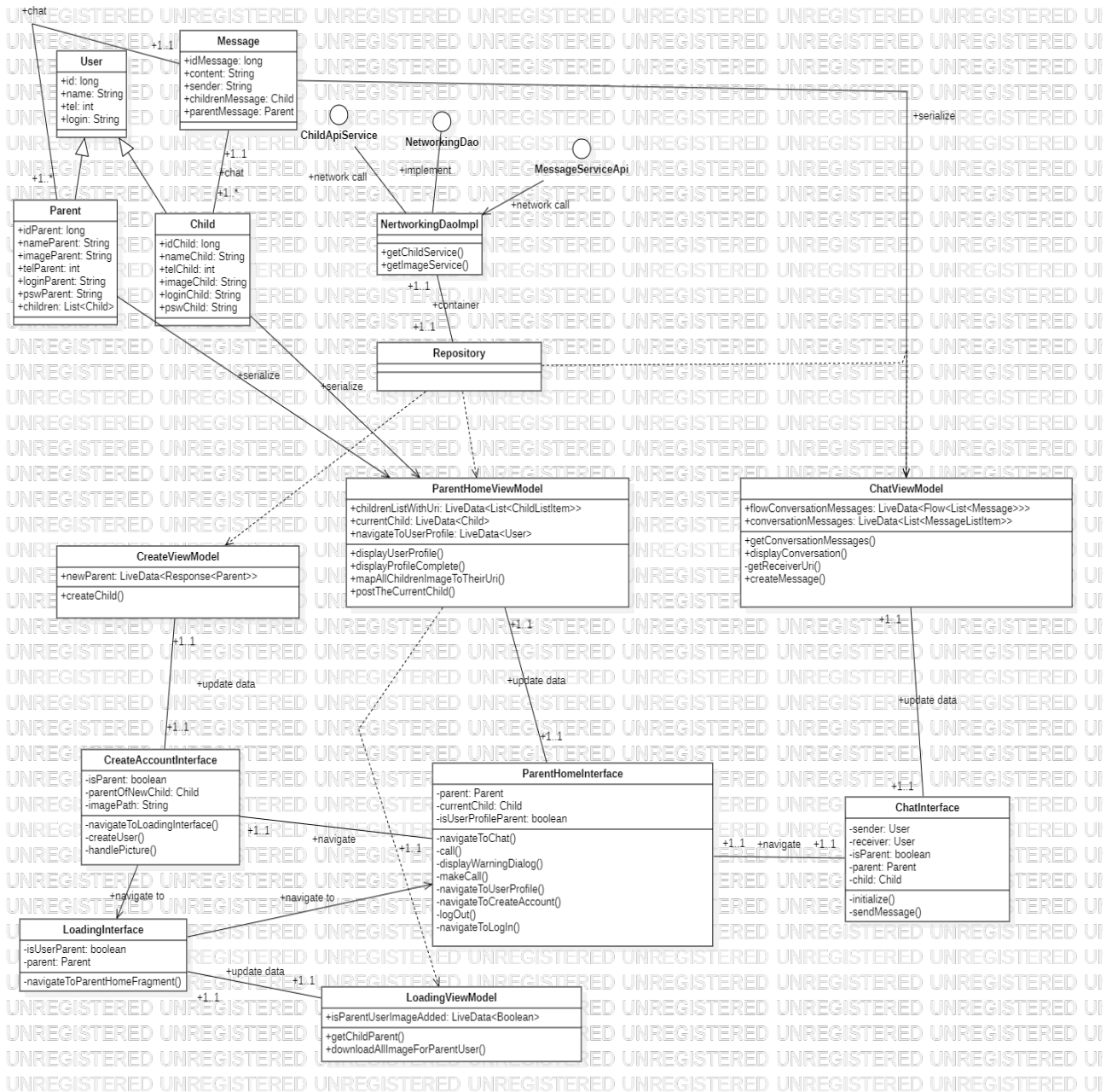


Figure 6.11: Sprint 2 class diagram

6.2 Production:

In order to show the results of this sprint, we will expose in this section the interfaces Man-Machine through the different screen prints produced.

6.2.1 Parent home interface:

The figure 5.12 show the parent interface at first launch.

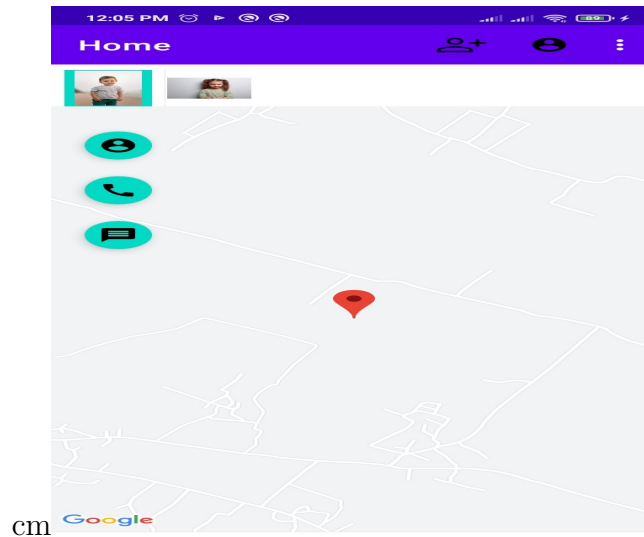


Figure 6.12: Parent home interface default

The figure 5.13 show parent home when parent select on newly added child. The figure

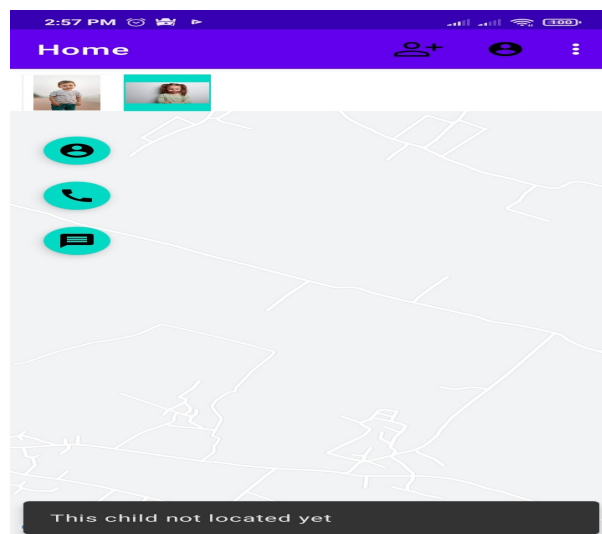


Figure 6.13: Parent home interface when newly added child is selected

5.14 show parent home interface when user call the selected child.

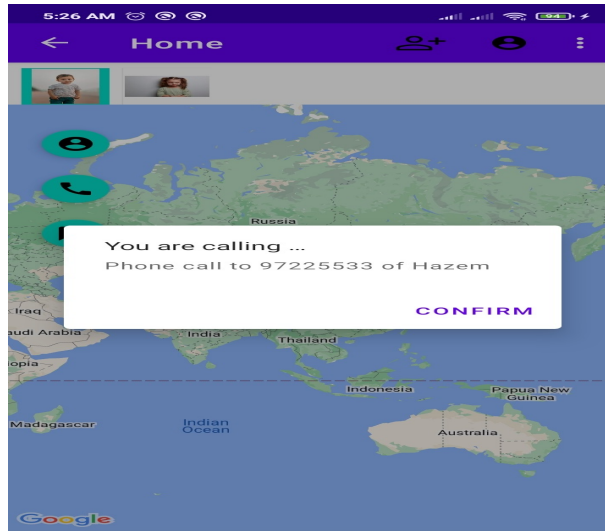


Figure 6.14: Parent home interface when parent call a selected child

6.2.2 Profile interface:

The figure 5.15 show parent profile.

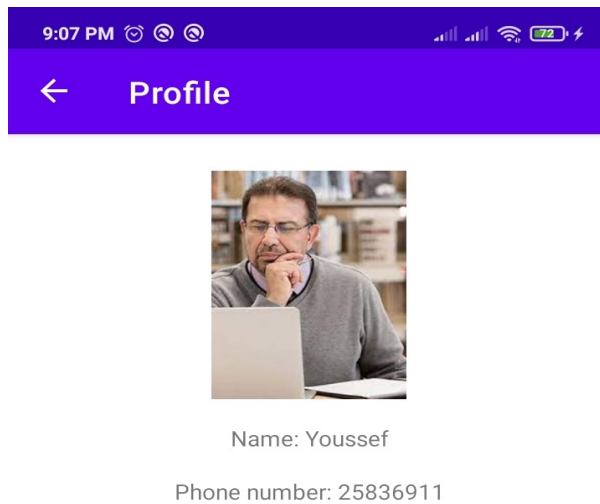


Figure 6.15: Parent profile interface

The figure 5.16 show profile of the selected child.

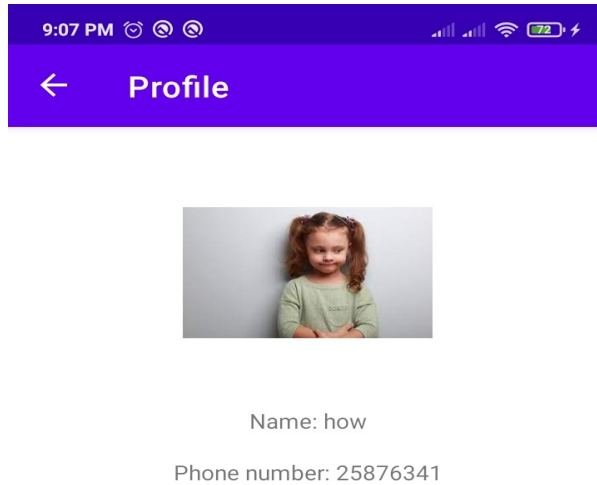


Figure 6.16: selected child profile interface

6.2.3 Chat interface:

The figure 5.17 show chat interface when the parent is the user of the app.

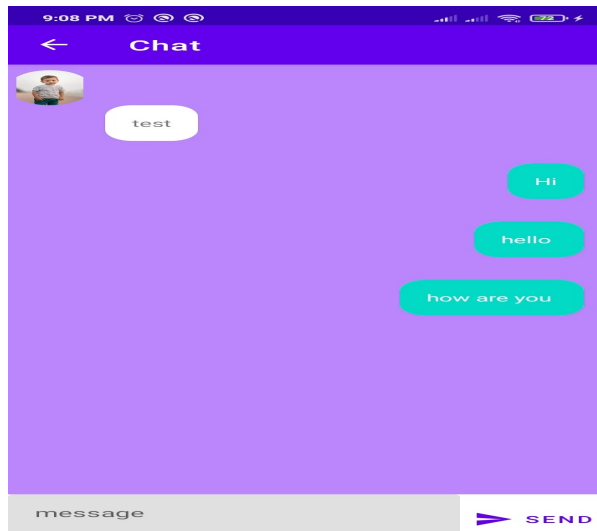


Figure 6.17: chat interface

6.3 Conclusion:

In this chapter, we have described the steps carried out during the second sprint which was devoted to parent interface. We started with a functional specification, then we have detailed the design by presenting the sequence, activity and class diagrams. Finally, we ended with

the presentation of the realization via the Man-Machine interfaces. The next chapter is devoted to the second release dedicated to the Data manipulation.

Chapter 7 Sprint3: Child interface:

This chapter is the subject of a presentation of the third sprint of the project which is the realization of the app accessible to the child users. the study of the analysis of the sprint, the conception, and realization.

7.1 Functional Specification:

7.1.1 Sprint Backlog:

After setting the goal of sprint, we define which "user stories" will be the subject of the Backlog for this sprint. We present through table 7.1 the different tasks of each user story.

User stories	priority
localization	100
Chat with the parent	80
Ask for emergency help	90
call the parent	70
see his profile	60
log out	50

Table 7.1: sprint 3 sprint backlog

7.1.2 Sprint detailed use case diagram :

We then present the analysis phase of which we describe the use case diagram and the textual description. Figure 7.1 describes the detailed use case diagram of the sprint as well as its textual description in table 7.2

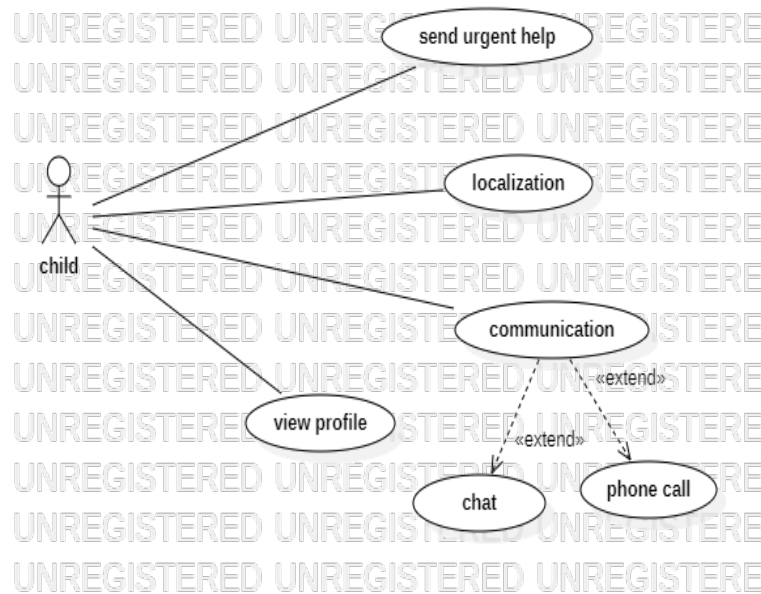


Figure 7.1: Sprint 3 detailed use case diagram

Clause	Description
Identify	Child functionality.
Summary	Child provide his location for the parent to track.
Actors	Child
Trigger context	Authenticated child
Pre conditions	parent related images downloaded and grand access location permission for the app
Description	When the child is authenticated , his related images has been downloaded and he give permission to access his location for the app he can access the childe features of the app

Table 7.2: Text description of sprint 3 use cases

7.1.2.1 communication use case diagram:

Figure 7.2 illustrates the diagram detailing the use case for communication as well as its textual description presented in table 7.3

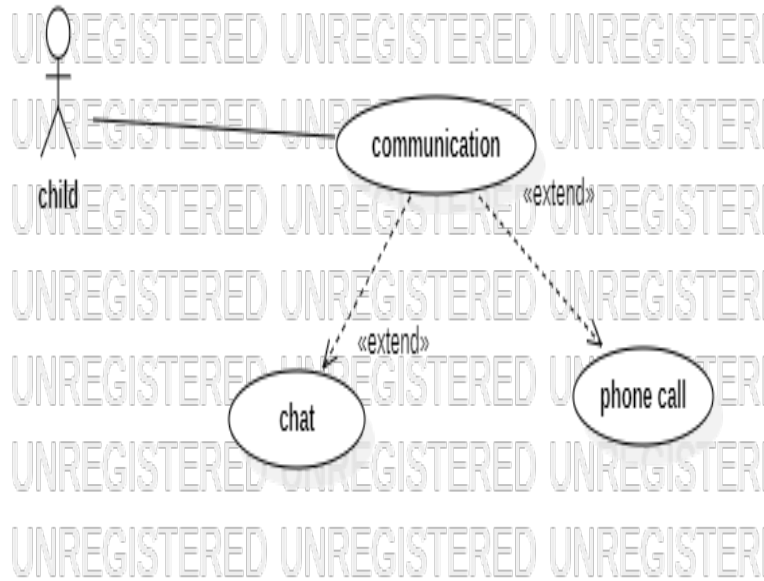


Figure 7.2: communication use case diagram

Use case	Communication
Actor	Parent.
Pre-condition	child click on messaging or call button.
Post Condition	Call has been made or chat interface will be displayed.
Nominal scenario	<ul style="list-style-type: none"> • Process the parent data. • Click on the chat or the call button. • Make the phone if the call what is been requesting. • Display The chat interface for messaging. • Fetch the old messages and display them. • Child can write his messages to his parent.

Table 7.3: Text description of Communication use cases

7.1.3 Conception:

In this part, we move on to the second activity in the product management sprint. variables which is the design where we present the diagrams of sequences, activities as well than the class diagram for this sprint.

7.1.3.1 Sequence diagram:

Sequence diagrams document the interactions to be implemented between objects temporally to achieve an outcome, such as a use case. The diagram of sequence lists objects horizontally, and time vertically. It models the execution different messages depending on the time.

7.1.3.1.1 Profile Sequence child: The figure 7.3 show how profile work. After child click on the profile icon in top bar ,he will be considered as the user to be display its profile and to be recognized to be as child.so his profile will be displayed and he will navigate to the profile interface who will look for his uri image in the android file system the show his profile

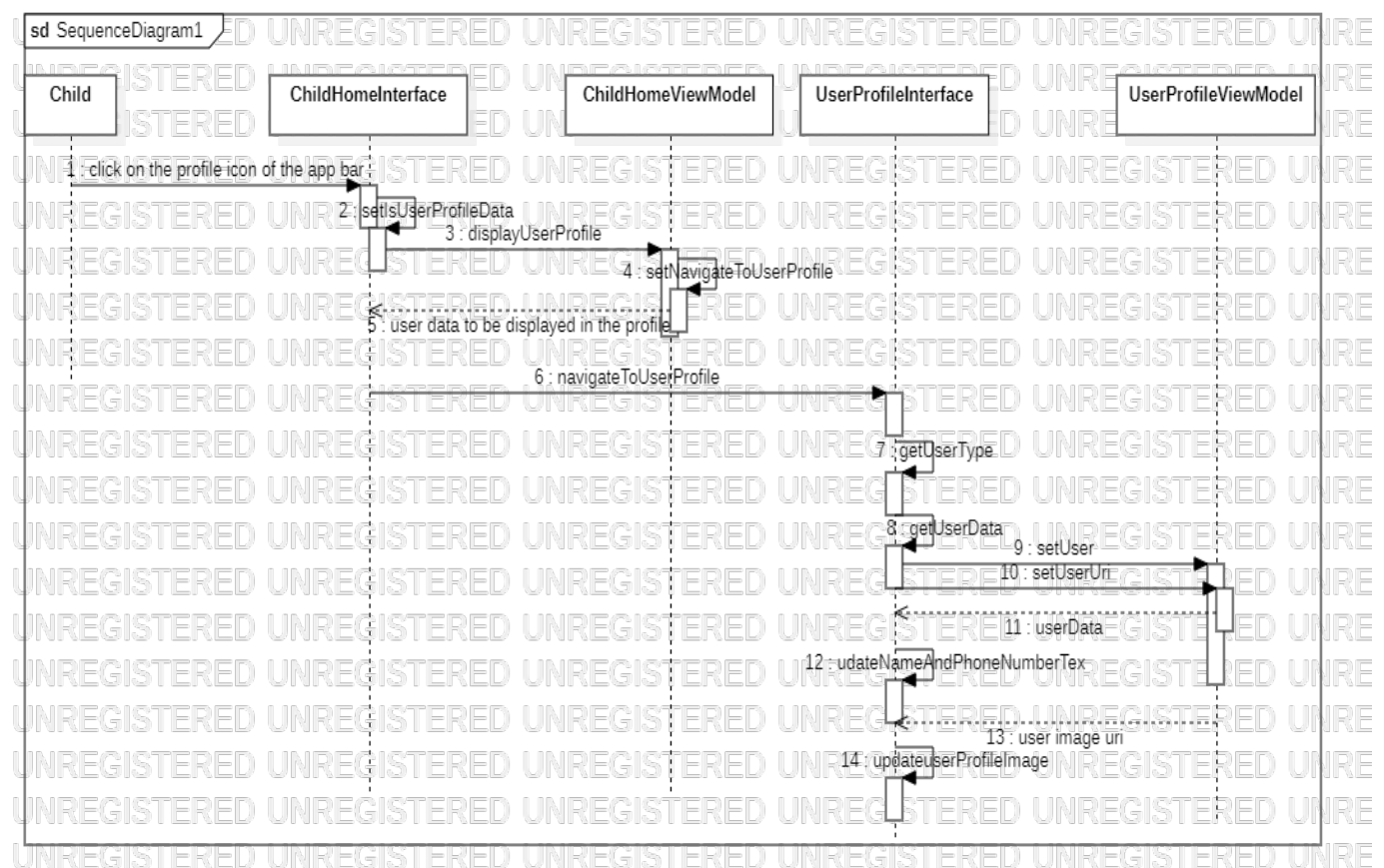


Figure 7.3: profile sequence diagram

7.1.3.1.2 Call Sequence Diagram: The figure 7.4 show how child can call his parent. first he need to grant permission or the app wont be able to make the call,if he grant the permission then he wont be asked for permission again unless he go to the android setting for the app and disable the permission.

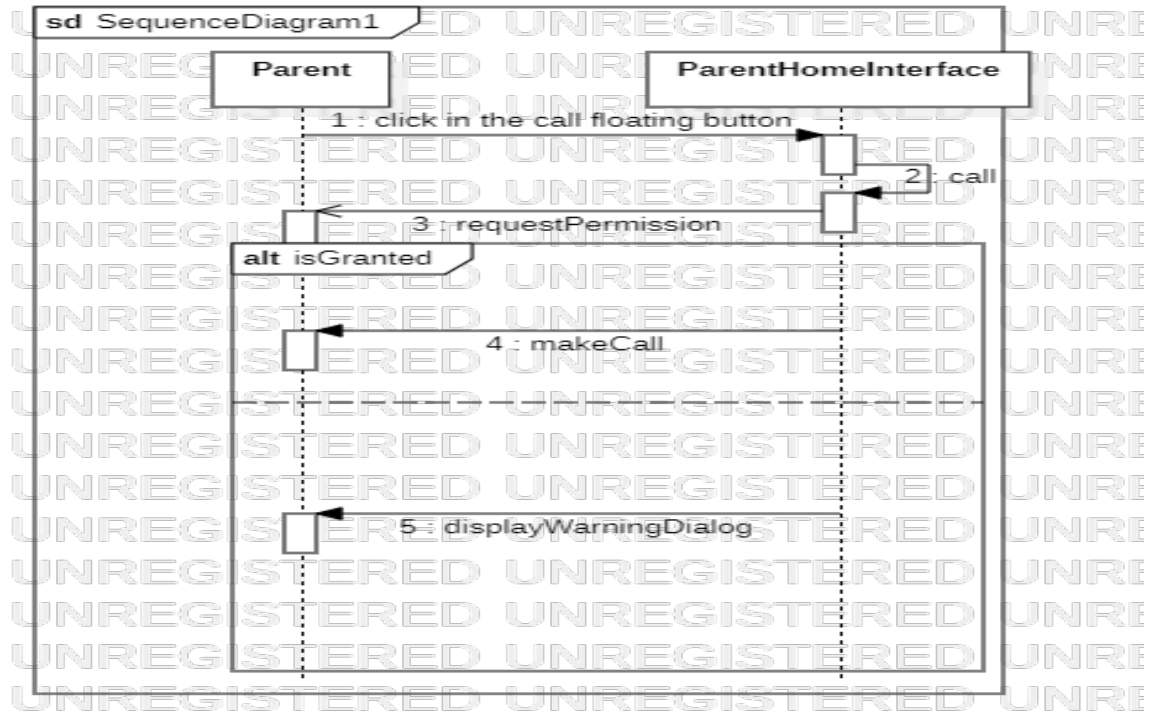


Figure 7.4: Call sequence diagram

7.1.3.1.3 Chat sequence diagram: The figure 7.5 show how child can chat with their child. The user(child) need to click on the chat button to start messaging their parent.

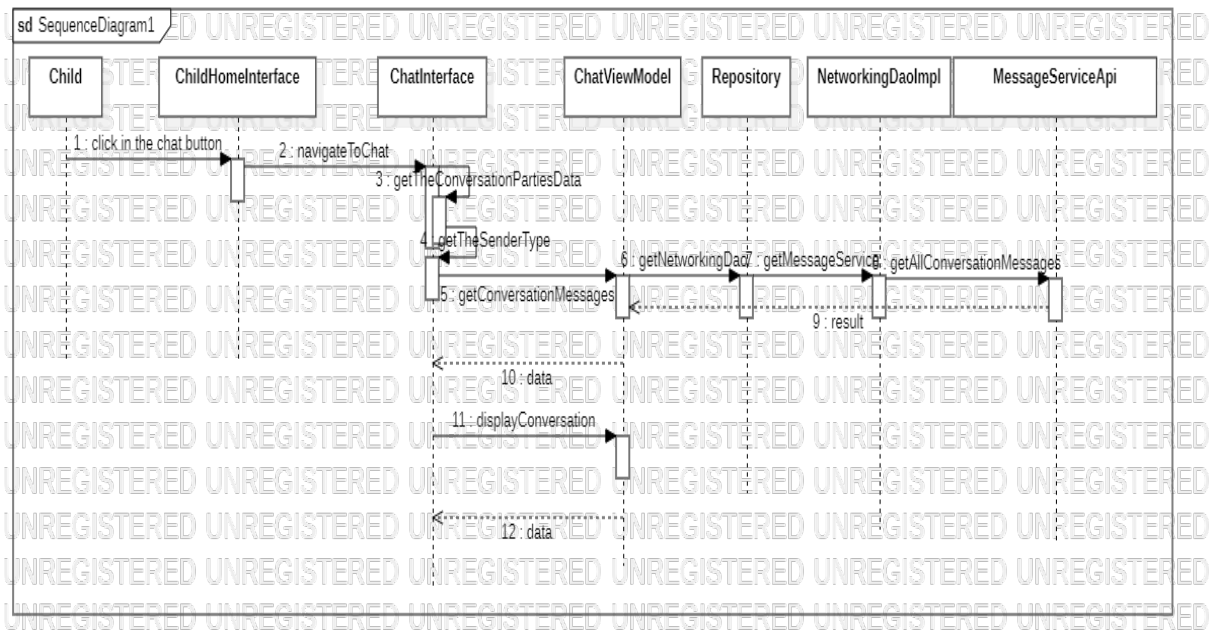


Figure 7.5: Chat sequence diagram

7.1.3.1.4 Urgent help sequence diagram: The figure 7.6 show how child can send urgent help. When child click urgent help button an SMS message sent to his parent that his child asking for help.

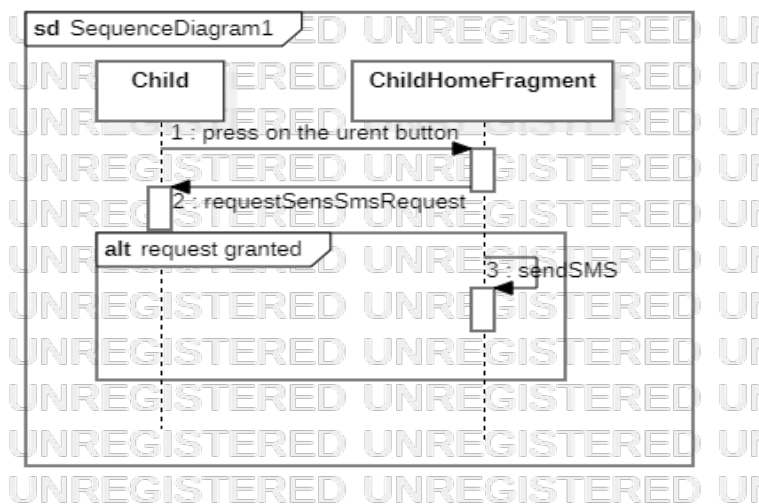


Figure 7.6: Urgent help sequence diagram

7.1.3.1.5 Localization sequence Diagram: The figure 7.7 show localization sequence diagram. as soon the child home interface start it will look for the child location and post it to the server,then when the child ask for his location the system will displayed for him.

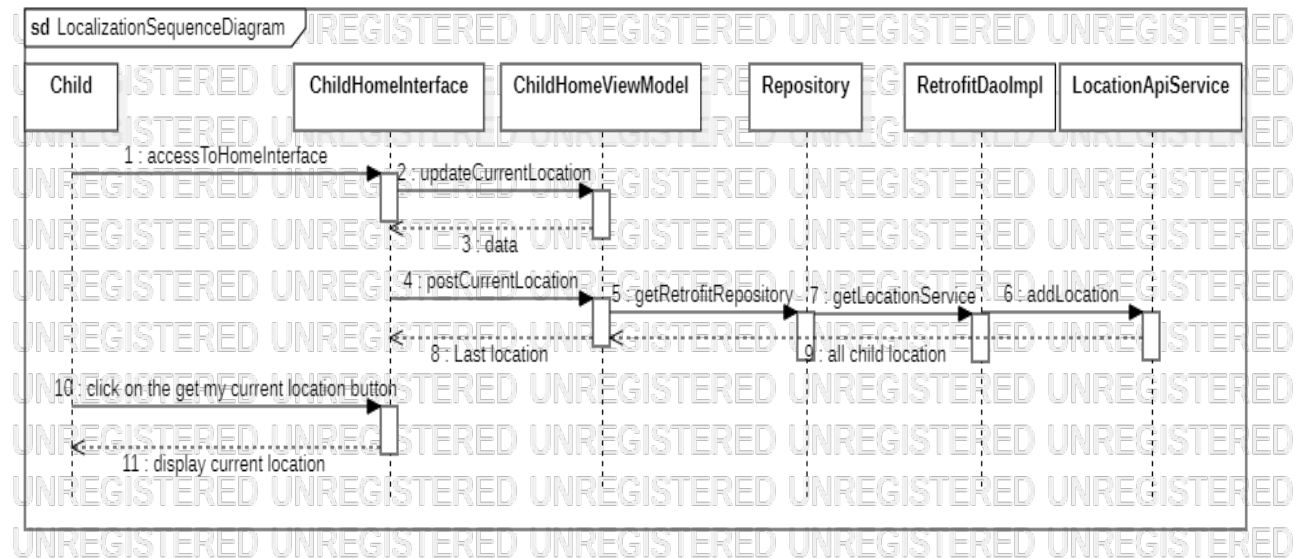
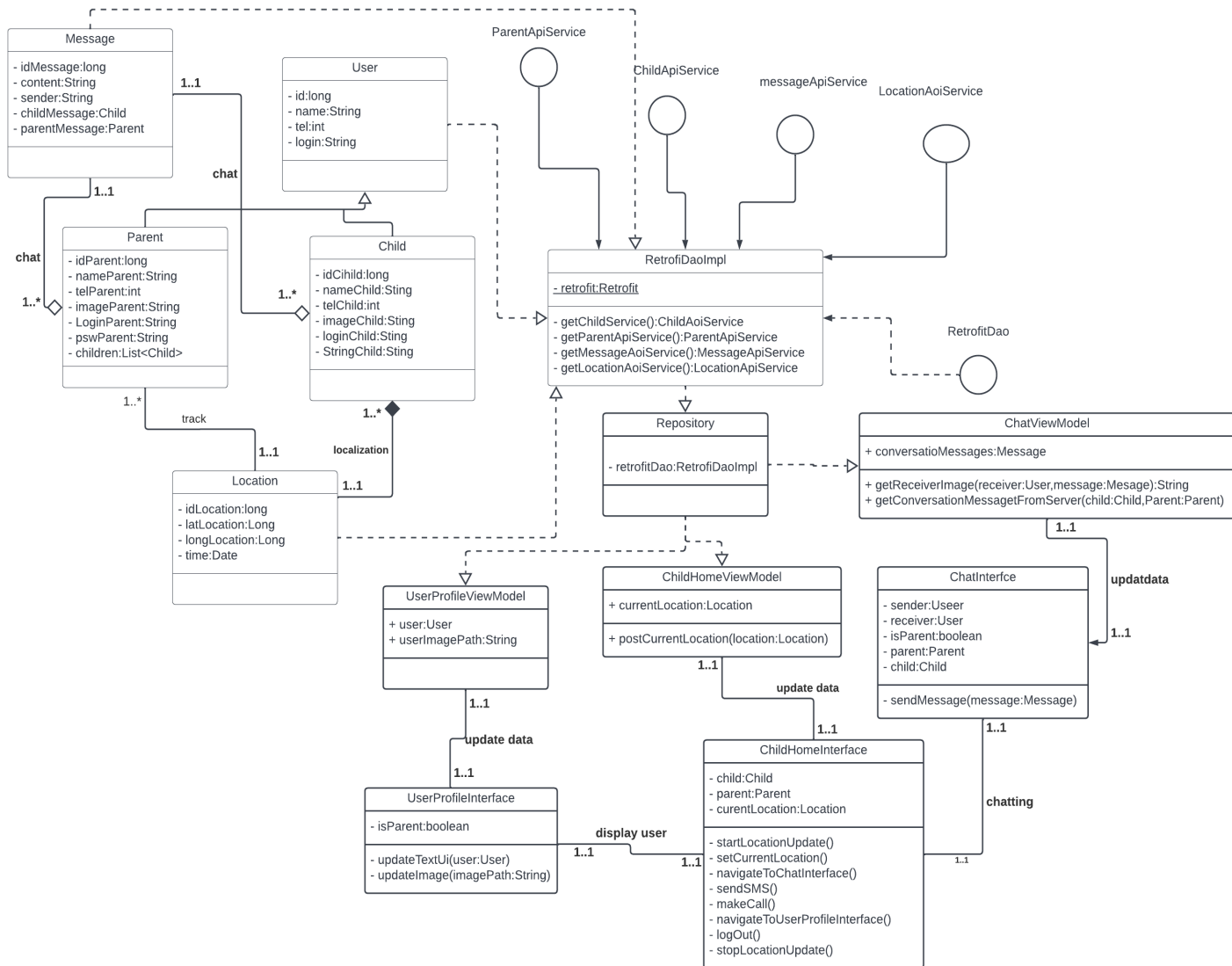


Figure 7.7: localization sequence diagram

7.1.3.2 Class diagram:

The class diagram makes it possible to describe the internal structure while showing the different classes, their attributes as well as the different structural relationships between these classes. It's about from a static view, because we do not take into account the time factor in the behavior of system. The class diagram models the concepts of the application domain as well as the internal concepts created from scratch as part of the implementation of an application. Each Object-Oriented Programming language gives a specific way to implement the paradigm object (pointers or not, multiple inheritance or not, etc.), but the class diagram allows you to model the classes of the system and them relationships independently of a programming language particular. Figure 6.8 describes the class diagram we used to develop the sprint.



7.2 Production:

7.2.1 home interface:

The figure 7.9 show home interface for child. After he successfully authenticated and all the related images to him downloaded the system will display this interface.

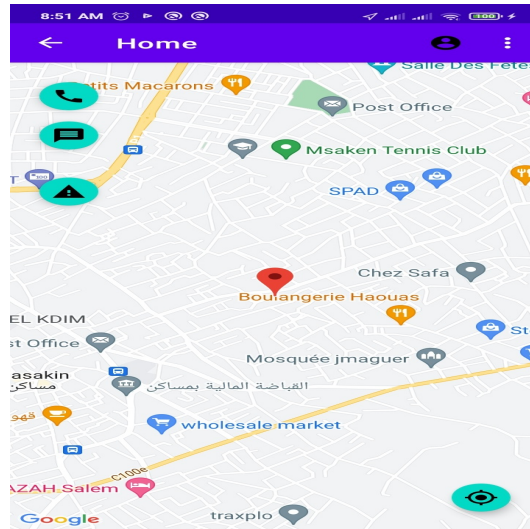


Figure 7.9: home interface

7.2.2 Profile interface:

The figure 7.10 show the user profile(child), he can access it after pressing the profile button in the action bar.

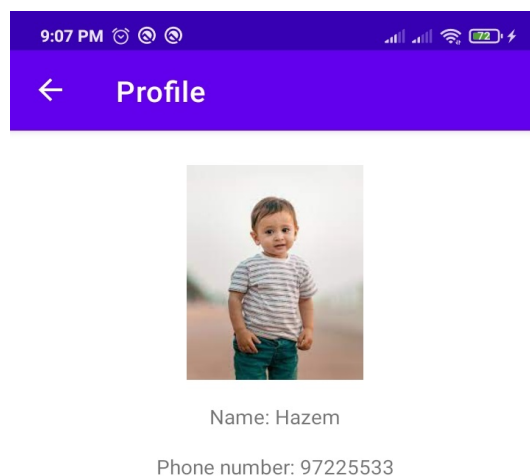


Figure 7.10: profile interface

7.2.3 Chat interface:

The figure 7.11 show the chat interface when the user click on the chat floating button.

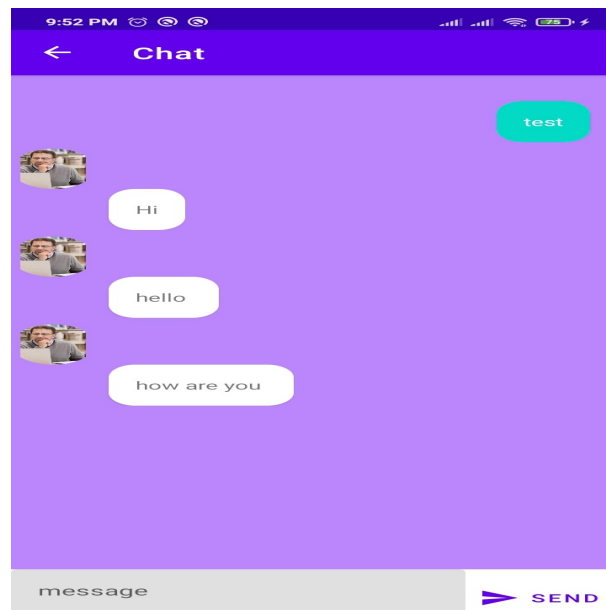


Figure 7.11: chat interface

7.3 Conclusion:

In this chapter, we have described the steps carried out during the third sprint which was devoted to child interface. We started with a functional specification, then we have detailed the design by presenting the sequence, activity and class diagrams. Finally, we ended with the presentation of the realization via the Man-Machine interfaces. The next chapter is devoted to the second release dedicated to the Data manipulation.

Webographie

- [1] <https://staruml.io/>,consulted on 02/03/2022 to 07/03/2022
- [2] <https://developer.android.com/studio>
- [3] <https://www.jetbrains.com/idea/download/?source=googleandmedium=cpcandcampaign=973696463>
- [4] <https://www.mysql.com/downloads/>
- [5] <https://spring.io/projects/spring-boot>,consulted on 02/03/2022 to 02/04/2022
- [6] <https://www.texstudio.org/>consulted on 02/03/2022 to 10/2022
- [7] <https://www.postman.com/api-documentation-tool/>,consulted on 20/03/2022 to /25/02/2022
- [8] <https://stackoverflow.com/>,consulted on 02/03/2022 to 25/05/2022
- [9] <https://github.com/>,consulted on 05/03/2022 to 15/03/2022
- [10] <https://medium.com/>,consulted on 03/04/2022 to 20/06/2022.
- [11] <https://developer.android.com/>,consulted on 02/04/2022 to 29/06/2022

General Conclusion:

The main objective of this internship is to develop an android application to maintain kids security,with giving parent permanent and immediate update about their kids location and allow them to communicate with kids.And their kids can communicate as well to ask their parent held in time of needs and provide the their current location in a map to help them if they lost In this report, we detailed the various stages of design and realization of the back office that we have developed. This end-of-study internship was the subject of an interesting experience, which allowed us to discover the professional world and to put into practice what we have acquired during our course and to improve our knowledge and skills in terms of technical. With more in mind to do like make our system help parent in observing the kids security, with the ability to identify dangerous and safe places, add ability for parent and child in exchanging pictures and add reward system for the kids

Summary:

This report presents the report of six months of internship for the end-of-studies project for obtaining the national computer engineering diploma. The objective of our project is to design and implement a solution whose purpose is to provide not only a locator app but also easy to use with a good experience and add little of bit of fu in using it.

Key word: Android, Spring boot ,My SQL, Kotlin, Retrofit